Luís Manuel dos Santos Gomes

Mestre em Engenharia Informática

Translation Alignment and Extraction Within a Lexica-Centered Iterative Workflow

Dissertação para obtenção do Grau de Doutor em

Informática

Orientador: José Gabriel Pereira Lopes,
Investigador Principal Aposentado,
Universidade Nova de Lisboa

Júri

Presidente:	Doutor José Júlio Alves Alferes
Arguentes:	Doutor Pablo Gamallo Otero
	Doutora Irene Pimenta Rodrigues
Vogais:	Doutora Maria Teresa Rijo da Fonseca Lino
	Doutor António Manuel Horta Branco
	Doutor Joaquim Francisco Ferreira da Silva
	Doutor José Gabriel Pereira Lopes

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Dezembro, 2017

Translation Alignment and Extraction
Within a Lexica-Centered Iterative Workflow

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "unlthesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/unlthesis [2] http://www.di.fct.unl.pt

Acknowledgements

First and foremost, I want to thank my advisor, Professor José Gabriel Pereira Lopes, for the excellent advice all these years, endless patience and encouragement. Surely, I have benefited from his long experience of advising PhD students, and for that I thank all of them (particularly António Ribeiro who has also worked on translation alignment and extraction, and whose work I have been extending since my MSc thesis).

A very special thanks to Professor Joaquim Ferreira da Silva and my longtime colleague José Aires for spending much of their time listening to my ideas and making insightful comments and suggestions. I believe that it is very important to have people willing to endure listen to our ideas *while they are still not mature*. I was very fortunate to have three persevering listeners: Professor Gabriel Pereira Lopes, José Aires and Professor Joaquim Ferreira da Silva. In the process of explaining an unripe idea, it always becomes clearer in our minds and often we realize some important detail that we overlooked earlier.

A special thanks to Kavitha for our productive and friendly collaboration, which I look forward to continue.

I thank the Portuguese Foundation for Science and Technology[1] for the financial support through individual doctoral grant[2], the ISTRION project[3] and the NOVA-LINCS laboratory[4] for supporting my work and the resulting scientific communications.

Thanks to my dear parents for their love, support, encouragement and understanding. I owe you so much more than I could express by words.

A big thanks to my mother in law, Maria Graciete, for all the encouragement and support, and for inspiring our family with her lively lifestyle – I wish I'll have the same energy when I get there.

Finally, a special thanks to my beloved Ana Luísa for all the support and patience, and for making me a better person; thanks to my dear daughter, Maria Leonor, for filling my life with happy and precious moments; and thanks to Maio, who welcomes me home every single day, wagging his tail like crazy...

[1] Fundação para a Ciência e a Tecnologia (`http://www.fct.pt`)
[2] Reference SFRH/BD/65059/2009
[3] Reference PTDC/EIA-EIA/114521/2009
[4] Reference UID/CEC/04516/2013 (`http://nova-lincs.di.fct.unl.pt`)

Abstract

This thesis addresses two closely related problems. The first, *translation alignment*, consists of identifying bilingual document pairs that are translations of each other within multilingual document collections (document alignment); identifying sentences, titles, etc, that are translations of each other within bilingual document pairs (sentence alignment); and identifying corresponding word and phrase translations within bilingual sentence pairs (phrase alignment). The second is *extraction* of bilingual pairs of equivalent word and multi-word expressions, which we call *translation equivalents* (TEs), from sentence- and phrase-aligned parallel corpora.

While these same problems have been investigated by other authors, their focus has been on fully unsupervised methods based mostly or exclusively on parallel corpora. Bilingual lexica, which are basically lists of TEs, have not been considered or given enough importance as resources in the treatment of these problems. Human validation of TEs, which consists of manually classifying TEs as correct or incorrect translations, has also not been considered in the context of alignment and extraction. Validation strengthens the importance of infrequent TEs (most of the entries of a validated lexicon) that otherwise would be statistically unimportant.

The main goal of this thesis is to revisit the alignment and extraction problems in the context of a lexica-centered iterative workflow that includes human validation. Therefore, the methods proposed in this thesis were designed to take advantage of knowledge accumulated in human-validated bilingual lexica and translation tables obtained by unsupervised methods. Phrase-level alignment is a stepping stone for several applications, including the extraction of new TEs, the creation of statistical machine translation systems, and the creation of bilingual concordances. Therefore, for phrase-level alignment, the higher accuracy of human-validated bilingual lexica is crucial for achieving higher quality results in these downstream applications.

There are two main conceptual contributions. The first is the *coverage maximization* approach to alignment, which makes direct use of the information contained in a lexicon, or in translation tables when this is small or does not exist. The second is the introduction of *translation patterns* which combine novel and old ideas and enables precise and productive extraction of TEs. As material contributions, the alignment and extraction methods proposed in this thesis have produced source materials for three lines of research, in the context of three PhD theses (two of them already defended), all sharing with me the supervision of my advisor. The topics of these lines of research are statistical machine translation, algorithms and data structures for indexing and querying phrase-aligned parallel corpora, and bilingual lexica classification and generation. Four publications have resulted directly from the work presented in this thesis and twelve from the collaborative lines of research.

Keywords: Document Alignment; Sentence Alignment; Phrase Alignment; Hierarchical Alignment; Discontiguous Phrases; Coverage Maximization; Bilingual Lexica Extraction; Spelling Similarity; Cognate Identification; Translation Patterns.

Resumo

Esta tese aborda dois problemas relacionados. O primeiro, *alinhamento de traduções*, consiste em identificar pares bilingues de documentos que sejam tradução um do outro numa colecção multilingue de documentos (alinhamento de documentos); identificar pares de frases que sejam tradução uma da outra em documentos alinhados (alinhamento de frases); e identificar expressões que sejam tradução uma da outra em frases alinhadas (alinhamento sub-frásico). O segundo é a extracção de pares de expressões equivalentes, a que chamamos *equivalentes de tradução* (ETs), a partir de corpora paralelos alinhados ao nível da frase e ao nível sub-frásico.

Estes problemas têm sido investigados por outros investigadores, mas sempre com foco em métodos não supervisionados e utilizando principalmente ou exclusivamente corpora paralelos. Léxicos bilingues, que no essencial são listas de ETs, não têm sido considerados ou, quando são, não lhes é dada a importância merecida na resolução destes problemas. A validação humana de ETs, que consiste na classificação destes como correctos ou incorrectos, também não tem sido considerada no contexto de alinhamento e extracção. A validação reforça a importância de ETs pouco frequentes (que constituem a maioria dos ETs de um léxico) aos quais é atribuído pouco peso por modelos estatísticos.

O objectivo principal desta tese é revisitar os problemas de alinhamento e extracção no contexto de um ciclo iterativo de trabalho que inclui validação humana. Por conseguinte, os métodos propostos nesta tese foram concebidos com vista a tirar partido de conhecimento acumulado em léxicos bilingues validados, bem como em tabelas de tradução obtidas por métodos não supervisionados. O alinhamento sub-frásico é um passo basilar para várias aplicações, onde se incluem a extracção de equivalentes de tradução, a criação de sistemas de tradução automática estatística e a criação de concordances bilingues. Portanto, em matéria de alinhamento sub-frásico, a precisão dos léxicos bilingues manualmente validados é decisiva para a obtenção de resultados com qualidade superior em todas as aplicações que destes alinhamentos dependem.

Ao nível conceptual, esta tese apresenta duas contribuições principais. A primeira é uma abordagem de *maximização de cobertura* como solução para os problemas de alinhamento. Esta abordagem faz uso directo do conhecimento contido num léxico ou numa tabela de tradução, caso não exista um léxico com tamanho suficiente. A segunda é a introdução dos *padrões de tradução*, que combinam uma série de ideias novas e antigas, e que permitem uma extracção precisa e produtiva de equivalentes de tradução. A um nível material, os métodos propostos nesta tese produziram alinhamentos e léxicos que serviram de base a três linhas de investigação, no contexto de três teses de doutoramento (duas delas já defendidas), partilhando comigo a supervisão do meu orientador. Os tópicos destas linhas de investigação são a tradução automática estatística, algoritmos e estruturas de dados para indexação e pesquisa sobre corpora paralelos alinhados ao nível sub-frásico, e classificação e geração de léxicos bilingues. Quatro publicações resultaram

directamente do trabalho apresentado nesta tese e outras doze resultaram das três linhas de investigação colaborativas.

Palavras-chave: Alinhamento de Documentos; Alinhamento de Frases; Alinhamento Subfrásico; Alinhamento Hierárquico; Expressões Não Contíguas; Maximização de Cobertura; Extracção de Léxicos Bilingues; Semelhança Ortográfica; Identificação de Cognatos; Padrões de Tradução;

Contents

List of Figures

List of Tables

Acronyms

AC Aho-Corasick algorithm, explained in Appendix A.

ANN artificial neural network.

CAT computer assisted translation.

DFA deterministic finite automaton.

EBMT example-based machine translation.

EdSim word similarity measure based on edit distance.

EM expectation maximization.

HPBSMT hierarchical phrase-based statistical machine translation.

ILP integer linear programming.

LCS longest common sub-sequence.

LCSR longest common sub-sequence ratio.

MT machine translation.

NMT neural machine translation.

PBSMT phrase-based statistical machine translation.

regex regular expression, explained in Appendix B.

SpSim language-pair-adapted word similarity measure that takes into account recurrent spelling differences such as "ph"↔"f" for English-Portuguese word pairs.

TE translation equivalent.

TectoMT machine translation based on tecto-grammatical tree representations of sentences.

TM translation memory.

Chapter One

Introduction

> Existing translations contain more solutions to more translation problems than any other available resource.
>
> Pierre Isabelle et al [49]

This thesis addresses the problems of translation *alignment* and *extraction* in the context of a lexica-centered iterative workflow that includes human *validation*. The main goal is to take advantage of accumulated knowledge, in the form of a list of bilingual pairs of word and multi-word expressions which have been manually classified by linguists as correct or incorrect translations. Hereafter, such a list will be referred to as a *bilingual lexicon*, and the pairs of word and multi-word expressions as translation equivalents (TEs).

Other authors address the alignment and extraction problems as fully unsupervised machine learning or mining tasks, with parallel and/or comparable corpora[1] as the main, and often the only source of information.

The ISTRION project [1] proposes a different perspective. One that introduces human validation of extracted TEs as a first level of supervision over the extraction of translation knowledge from parallel corpora. This perspective is grounded on the observation that even people need to be corrected from time to time. Why would a machine need no supervision at all?

The introduction of human validation of extracted TEs required the creation of bilingual lexica databases to persist, update and grow knowledge over time. These databases then became a central resource available to the extraction, alignment and translation processes, in addition to parallel corpora.

1.1 Context: A Lexica-Centered Iterative Workflow

This thesis results from an investigation that started in the context of the ISTRION project [1] and later continued in the context of the ISTRION-BOX startup company [50].

[1] Corpora is the plural of corpus, which is a collection of texts. A parallel corpus is a collection of texts and their translations, typically aligned at sentence level. Comparable corpora are collections of texts that are not necessarily translations of each other but address similar topics or domains.

Thus, I addressed alignment and extraction problems as part of a broader lexica-centered iterative workflow encompassing the following steps:

1. Alignment of documents retrieved from the web or from any other source;
2. Alignment of parallel[2] sentences within parallel documents;
3. Alignment of known translation equivalents (TEs) within parallel sentences;
4. Extraction of new TEs from aligned texts;
5. Machine classification of newly extracted TEs using machine-trained classifiers [69];
6. Human validation of extracted and pre-classified TEs.

These steps are depicted as large arrows in Figure 1.1, where the type of input and output data of each step is also illustrated.

The workflow is *interactive* because it involves human interaction in the validation step. It is also *iterative* because it is to be repeated over and over again.

Each iteration increases the size of the accumulated bilingual lexica, improves its coverage and correctness, and allows more fine-grained and accurate alignments in the next iteration. Subsequent extraction and classification steps, both benefit from the refined alignments (lower ambiguity) and the increased size of accumulated lexica (more examples to learn from). Furthermore, since the phrase-aligned parallel texts and the accumulated bilingual lexica are used as the basis for creating phrase-based statistical machine translation systems (PBSMT), they also improve over time, although not linearly [4].

As depicted in Figure 1.1 the bilingual lexicon has a central role in this workflow. After extraction, TEs are inserted into the bilingual lexicon with status *unverified* (U). Later, they are manually validated by changing their status to either *accepted* (A) or *rejected* (R). The human evaluation is facilitated by a bilingual concordancer that incorporates sentence and phrase level alignments [23, 24, 26]. In cases where the evaluator is uncertain about a TE, he/she may postpone its validation.

The alignment and extraction steps should be designed in a way that takes advantage of the knowledge available in the human-validated lexicon. However, alignment and extraction methods previously proposed by other authors focus on parallel corpora as the primary source of information. Bilingual lexica, when considered, is relegated to a secondary level of importance. As a consequence, the performance of these methods does not improve significantly when a bilingual lexicon is given.

This thesis proposes a suite of alignment and extraction methods specifically designed to take advantage of and contribute to the long term growth of a manually-validated bilingual lexicon. These methods are the main contribution of this thesis and their development was the main goal.

[2]Parallel sentences are sentences that are translations of each other. Likewise, we may have parallel documents, paragraphs, phrases, etc.

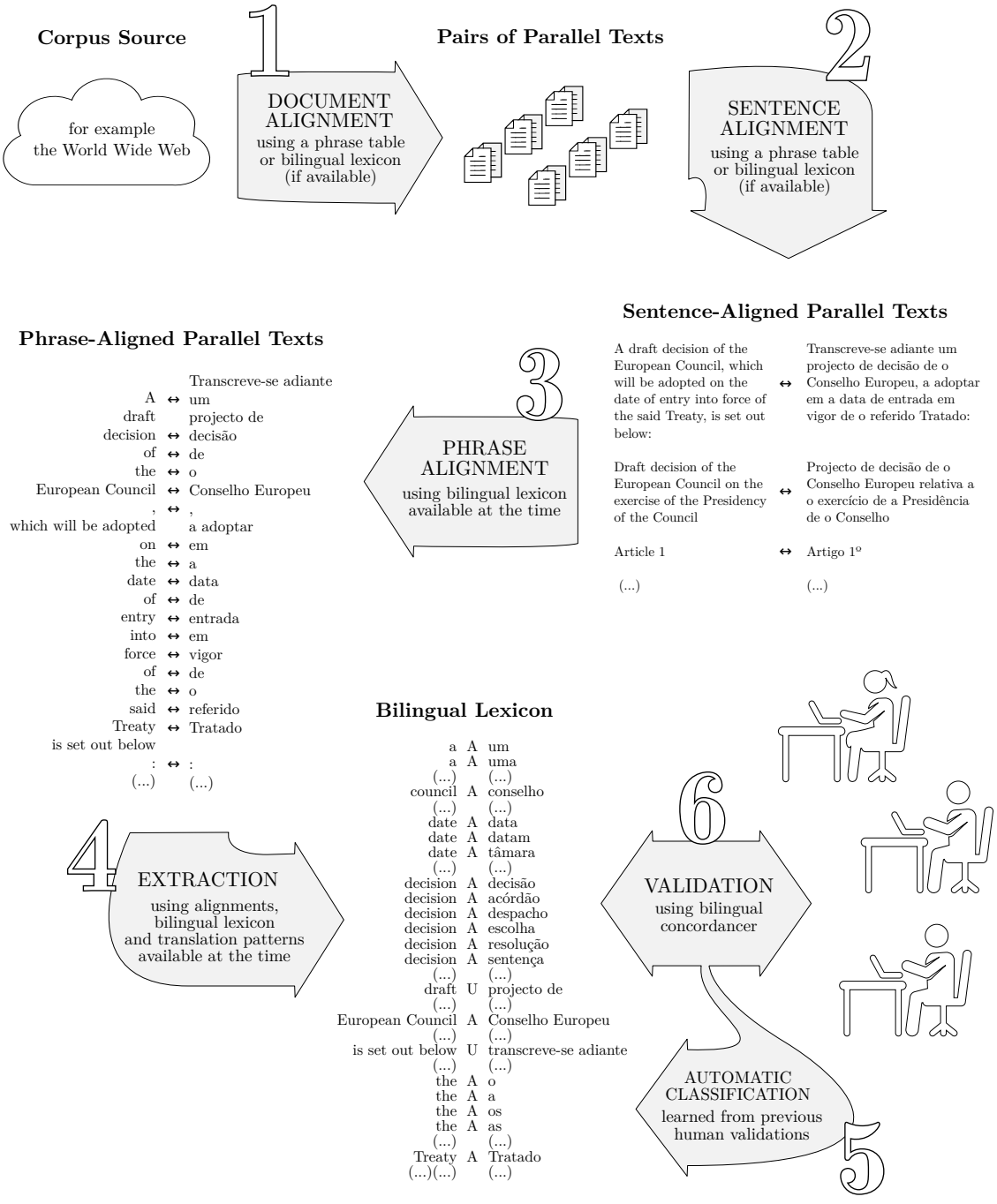


Figure 1.1: Global iterative workflow with lexica as a central resource and featuring human-validation.

1.2 Translation Alignment

Translation alignment, or just *alignment* hereafter, consists of identifying corresponding *alignment units*, which can be words, phrases[3], sentences, paragraphs, or any other text units, within an *alignment scope* defined as a pair of larger text units that are translations of each other.

For example, the problem of aligning phrases within parallel sentences is an alignment problem where the alignment units are phrases and the alignment scope are pairs of parallel sentences. Another example alignment problem is finding documents that are translation of each other within a multilingual collection of documents (multilingual corpus). Here, the alignment units are documents and the alignment scope is a collection of documents.

Table 1.1 lists the most commonly addressed alignment problems, characterized in terms of alignment units and scopes.

Alignment Problems

Unit	Scope	Commonly Adopted Designation	Sample Publications
word	sentence	word alignment	[16, 64, 113]
phrase	sentence	phrase alignment	[77, 117]
word/phrase	document	anchored alignment[4]	[28, 39, 48, 79, 95]
sentence	document	sentence alignment	[37, 59, 81, 99, 110]
document	corpus	document alignment	[17, 35]

Table 1.1: Alignment problems described in terms of aligment units and scopes.

The result of an alignment process[5] is a set of *alignment links* between pairs of alignment units within a given scope. This set is simply called an *alignment*.

Initially, an alignment process will consider several hypothetical alignment links, which we call *alignment candidates*, or just *candidates*. Afterwards, the alignment process will select a *consistent subset* of candidates to become the final set of alignment links, i.e. the resulting *alignment*.

This thesis addresses three alignment problems: document alignment in Chapter 2, sentence alignment in Chapter 3 and phrase alignment in Chapter 4. These problems correspond to three alignment steps needed to process a collection of documents downloaded from the web or from another source, down to the level of fine-graininess needed to train statistical machine translation systems, enable extraction of word and multi-word translation equivalents (TEs), and enable the creation of bilingual concordances.

[3] *Phrase*, as usual in the computational linguistics area, means a word sequence, contiguous or not.

[4] These methods do not have a commonly adopted designation. However, because all these algorithms rely on and produce *alignment anchors*, which will be explained later in Chapter 4, we refer to this class of problems as *anchored alignment* problems.

[5] We use the term *process* to refer to a running instance of a computer program.

1.3 Coverage

The alignment methods proposed in this thesis for the three distinct alignment problems addressed, are all based on variants of a general concept which we call *coverage*.

In general terms, *coverage* is a measure of how much information from a bilingual lexicon is consistent[6] with a given set of alignment links. The underlying hypothesis of the proposed coverage-based alignment methods is that the correct set of alignment links for a given problem instance should be the one, among all hypothetical sets of alignment links, that maximizes the coverage with respect to a bilingual lexicon.

Instead of a bilingual lexicon, we may compute coverage with respect to a bilingual phrase translation table, such as those employed in statistical phrase-based machine translation systems, obtained with unsupervised methods. Using a phrase translation table instead of a lexicon is an option in situations where there is no lexicon available, or it is too small. In any case, by using a bilingual lexicon or a phrase translation table, we are reusing previously obtained translation knowledge to align new translations.

Depending on the alignment units being considered in an alignment problem and the size of the available validated lexicon, the larger number of translations present in a typical phrase translation table obtained with unsupervised methods may counterbalance the higher accuracy but smaller size of the lexicon. For example, in our document alignment experiments we discovered that a phrase translation table with approximately 50 million phrase pairs, obtained with the Moses toolkit [55] over the Europarl corpus [53], despite containing many errors, yields marginally better alignments (1 percent higher recall) than when using a manually validated lexicon with approximately 400 thousand TEs.

However, for the phrase alignment method proposed in Chapter 4, phrase translation tables are not adequate at all. Not only because they contain a large number of incorrect translations, but more importantly, because the phrases contained in these tables are arbitrarly segmented and overly redundant, which would greatly increase the computational cost of phrase alignment as well as decrease its accuracy and usefulness.

1.4 Translation Extraction

Translation extraction, or just *extraction* if no confusion arises, consists of extracting a list of bilingual pairs of hypothetically equivalent word and multi-word expressions, which we call *translation equivalents* (TEs). Human validation of TEs consists of classifying them either as *accepted* or *rejected*. Extracted TEs that have not yet been validated are implicitly classified as *unverified*. A set of TEs is a *bilingual lexicon* or just *lexicon* if no confusion arises. When describing an alignment or extraction method that takes advantage of apriori knowledge in the form of an input bilingual lexicon, we say that a TE is *new* or

[6] Later, we will be more precise about what we mean by "consistent".

unknown if it is not (yet) in the lexicon when the method starts being executed. Conversely, TEs are *known* if they are in the lexicon at that point in time.

Like alignment, extraction may be performed at different scopes and targetting different *extraction units*. We characterize extraction problems by the *co-occurrence scope* and the *global scope*, which may be same in some cases. The co-occurrence scope is determined by the type of parallel segments that are given as input to an extraction method. As the name implies, this scope determines the co-occurrence of bilingual pairs of expressions. The most commonly considered co-occurrence scope is the sentence, which means that a pair of expressions appearing in parallel sentences will be considered to co-occur, independently of their position within the sentences. Finer or coarser co-occurrence scopes are possible, such as phrase, paragraph or document.

The *global* extraction scope refers to the type of collection of co-occurrence scopes that is provided to the extraction method. The two most commonly considered global scopes are the document and the corpus.

Statistical extraction methods typically require a large corpus as global scope in order to have reliable statistics.

The two extraction methods presented in this thesis take advantage of previously validated TEs to extract new TEs with high precision, even if they occur only once or twice in the extraction scope. Thus, unlike statistical methods, my proposed methods may be applied to a single pair of documents as well as a corpus.

1.5 Alignment versus Extraction

Alignment and extraction are very closely related, and sometimes confused with each other[7], but are complementary operations, to some extent. They are perhaps better understood when compared in terms of their inputs and outputs:

- An alignment process takes as input a pair of parallel texts, the *alignment scope*, and produces a set of *alignment links* between *alignment units* within that scope.

  For example, a sentence aligner is a process that takes a pair of parallel documents as input, and produces a list of pairs of sentence groups that should be equivalent. If two imaginary documents A and B had 3 and 4 sentences, respectively, then a possible sentence alignment would be [(1,1),(2,2-3),(3,4)]. Translated to plain English, this alignment means that the first sentences of both documents align with each other, the second sentence of A aligns with the second and third sentences of B and the last sentences of both documents align with each other.

  In the case of the alignment methods proposed in this thesis, the alignment process also takes as input a bilingual lexicon, which may be fully or partially human-validated, and which carries our accumulated knowledge.

[7] For example, Anymalign [60] is referred to by its authors as an aligner, but it is in fact an extractor.

- The extraction process takes as input a set of aligned units and produces a list of candidate TEs. Typically, the extracted TEs are smaller than the aligned units given as input. The whole set of aligned units given as input is the *global scope* and each aligned unit is the *co-occurrence scope*.

1.6 Knowledge Persistence and Growth

Once a TE has been manually validated, it will never be removed from the lexicon. It may, however, change its status. We keep both the accepted and rejected TEs, because rejections are important knowledge too. Keeping rejected entries allows us to avoid repeating the same extraction mistakes. Furthermore, using previously accepted and rejected TEs as examples, Kavitha Mahesh [69] trained automatic classifiers which were then used to pre-classify newly extracted TEs as *pre-accepted* or *pre-rejected* before human validation. Pre-classification boosts human validation productivity by greatly reducing the number of mouse movements and clicks needed to validate a full page of TEs.

Therefore, once acquired, knowledge *persists* over time. Moreover, as we continue to align new parallel texts, extracting and validating new translation equivalents, knowledge *grows* over time. By contrast, models obtained by machine learning approaches are typically discarded and retrained from scratch whenever we change from one corpus to another, which leads us to call these approaches *memoryless*.

In our lexica-centered approach, knowledge is encoded in an explicit, unambiguous, human-friendly and future-proof format. A lexicon is, conceptually, a long table with at least three required columns: a pair of word or multi-word expressions and a status label which can take one of three values, *accepted* (A), *rejected* (R) or *unverified* (U), as shown down at the bottom center of Figure 1.1.

Since the multiple processes composing the workflow (alignment, extraction, validation, etc) are typically distributed over a newtork of computers, each one keeps a local copy of the lexicon, for efficiency. These copies are synchronized periodically with the *master* lexicon, which is held by the validation process.

Each specific process may enrich its local lexicon table with additional columns to serve its own needs. For example, the master version, held by validation process, contains timestamps of when the TE was extracted and validated, a reference to the method that extracted the TE, a reference to the human validator, etc.

In its simplest archival form, the lexicon table is stored in a plain text file, using tabs as column separators, and all metadata stored by the validation process is kept.

The local copies used by each process, are stored in whatever format better serves the needs of that process. For example, the validation process stores the lexicon in a relational database, in order to efficiently query and manipulate the lexicon with SQL

statements[8]. A completely different process, such as the phrase alignment method proposed in Chapter 4, stores the lexicon in a specialized data structure that enables fast lookup of all known TEs within a pair of parallel texts.

In any case, the format and information contained in the master lexicon table can easily be adapted and consumed by any tool. Now or in the future.

By contrast, model parameters obtained by a machine learning software package are seldom transferrable to other similar software packages. Sometimes, models produced by one version of a software package are not readable by later versions of the same package. This encourages, or even forces, the memoryless practice of discarding previously trained models and retrain them anew, when new corpora needs to be processed.

1.7 Human-Validation Investment

The cost of creating unsupervised machine translation systems such as those based on Moses [55] is largely bound to the cost of the computational infrastructure needed to train and run the statistical or neural models.

However, the human effort needed to post-edit translations produced by these unsupervised systems will not decrease significantly over time, because these systems are relatively insensitive to small data increments. To be more concrete, let us consider a typical Moses [55] system trained on a corpus with hundreds of millions of tokens. Note that such a corpus size is not considered a "big corpus" in the context of PBSMT.

Let us now assume that after using this system for a few weeks or months, we have produced new high-quality manually post-edited translations. Then, hoping to see an improvement in the quality of future translations made by the system, we would add these translations to the original corpus and retrain the models. Unfortunately, it is unlikely that our effort would be rewarded because the quantity of new translations is too small in comparison with data used to train the first models. During a period of weeks or months, even the most productive human translator/post-editor will produce only a tiny amount of new translations when compared to the size of the original corpus (hundreds of millions of tokens). Due to the statistical nature of the system, a small change in the data is unlikely to produce a significant change in the output and in practice the improvements are barely noticeable. It is even possible that adding a small amount of data will result in a slight degradation of translation quality, as shown by Koehn et al [97]. While it is not clear why these fluctuations happen, one possible reason is that the training of translation models involves random sampling at several stages.

Ultimately, a fully unsupervised approach to let the machine learn how to translate is utopic because even children, who have far more sophisticated cognitive mechanisms than machines currently do, need interaction with adults to learn how to speak languages

[8] When stored in a relational database, the lexicon is in fact split and reorganized into several tables as a consequence of database normalization, to avoid redundancy and increase data integrity.

fluently and to translate from one language to another. Thus, our lexica-centered approach results from a conscious choice to include human input in the system, to enable a sustainable improvement across all of the connected technologies (alignment, extraction, machine translation, etc). As a result of this decision, the Transtor PBSMT[9] system [4] surpasses the state-of-the-art Moses [55] PBSMT system by 5.1 BLEU[10] points on average, for 8 language pairs[11] and 16 translation directions, when trained over a medium-sized corpus. When a small corpus is used to train both systems, the difference increases to almost 15 BLEU points, on average.

Transtor derives its phrase table from the phrase level alignments produced by the method proposed in Chapter 4, which in turn are based on a human-validated bilingual lexicon. Besides this extracted phrase table, Transtor also uses human-validated TEs from the lexicon directly.

For the English-Portuguese language pair, on which we have been working on for longer and for which we have a larger accumulated bilingual lexicon, Transtor achieves 29 BLEU points more than Moses, when both systems are trained on a small corpus.

Another motivation for our lexica-centered approach is the increasing demand for bilingual (and multilingual) lexica in the context of the translation industry, where these resources are *leveraged*[12] by CAT tools, reducing time and cost of translation.

To conclude, we should think about human-validation as an investment, rather than an expense.

1.8 Summary of Contributions

Considering the main goal of this thesis, which is to address the problems of translation alignment and extraction while taking advantage of accumulated knowledge in the form of human-validated bilingual lexica, this thesis contributes:

1. Three novel alignment methods, described in Chapters 2, 3 and 4. These methods are based on maximization of *coverage* measures, which, in general terms, are measures of how much of the information contained in a given bilingual lexicon or phrase translation table is consistent with an hypothetical alignment. The underlying hypothesis is that the correct alignment for a given problem instance should be the one, among all hypothetical alignments, that maximizes the coverage score.

[9] Phrase-Based Statistical Machine Translation

[10] BLEU is a commonly used translation quality metric proposed by Papineni et al [88]. While the metric ranges from 0 to 1, it is more commonly presented multiplied by 100, thus in the range 0 to 100. In this thesis, we also follow this convention.

[11] English-German, English-French, English-Spanish, English-Portuguese, German-Portuguese, Spanish-Portuguese, French-Portuguese and German-Spanish.

[12] *Leverage* is a commonly used verb, in the context of computer assisted translation (CAT), to denote usage of resources (such as bilingual lexica) in a way that boosts human translators productivity.

2. A cognate extraction metod, in Chapter 5, that makes use of human-validated TEs to extract substitution patterns such as ("ph", "f") from cognates such as “pharmacy”↔“farmácia”, which are then used to compute a spelling similarity measure. This similarity measure, in combination with a co-occurrence similarity measure computed over a parallel corpus aligned at sentence level, enables the extraction of equivalent cognates with higher precision than when other spelling similarity measures are used instead.

3. A more general extraction method, in Chapter 6, based on patterns that take morphology, close-range reorderings and context into account. This extraction method is able to extract hundreds of thousands of word and phrase translations with precisions ranging from 90% to 100%, depending on the patterns being used.

Besides the direct contributions listed above, the alignment and extraction methods proposed in this thesis have also produced alignments and bilingual lexica that have been used as the basis for further research in the context of three other PhD theses, two of them already defended.

Specifically, the phrase level alignments produced by the method proposed in Chapter 4 have been used by José Aires, in the context of his Phd thesis [4], to build phrase-based statistical machine translation systems for 8 language pairs[13] and 16 translation directions. These systems perform, on average, 5.1 BLEU points above the Moses system [55], when trained over a mid-sized corpus, specifically the DGT-TM [104]. When trained over the small OPUS euconst corpus [107], the performance difference dilates to 14.8 BLEU points on average.

We participated on the WMT16 biomedical translation task (competition), where we got independent confirmation [13] of the better performance of our PBSMT system in comparison to the Moses system. Our system was described in a co-authored paper [6], published in the WMT16 proceedings.

The phrase level alignments and their application for: (1) machine translation, (2) extraction of translation equivalents and (3) bilingual concordancing, motivated the work done by Jorge Costa in the context of his PhD thesis, which is focused on building and querying compressed indices for bilingual corpora aligned at phrase level. In this line of research, we have co-authored four papers [24, 25, 26, 27], published in conference proceedings.

Motivated by the translation patterns proposed in Chapter 6, Jorge Costa has developed an algorithm for finding occurrences of bilingual pairs of gapped phrases, aiming to support the full syntax and semantics of translation patterns in future work.

The bilingual lexica automatically extracted with several methods, including the ones proposed in Chapters 5 and 6, and which has been manually validated over the timespan

[13] EN-DE, EN-FR, EN-ES, EN-PT, DE-PT, ES-PT, FR-PT and DE-ES

of nearly a decade, has motivated and enabled the work done by Kavitha Mahesh in the context of her PhD thesis [69].

Kavitha Mahesh addressed two problems. The first one is to automatically classify newly extracted TEs as correct or incorrect, using a classifier trained on TEs previously validated by humans. The second one is to automatically generate translations for word forms that do not occur, or have not been extracted from aligned bilingual corpora.

In the context of our collaboration, we co-authored seven articles [70, 71, 72, 73, 74, 75, 76], published in conference proceedings.

Besides the abovementioned publications in collaboration with José Aires, Jorge Costa and Kavitha Mahesh, the methods proposed in Chapters 2, 3 and 5 have been published separately in conference proceedings [41, 42, 43].

1.9 Structure of this Thesis

The next three chapters (2, 3 and 4) address three alignment problems of increasingly finer-grained alignment units: document, sentence and phrase.

Chapters 5 and 6 address translation extraction, exploiting different kinds of similarity between translations. While in Chapter 5 we exploit the spelling similarity of cognate words belonging to distinct languages, in Chapter 6 we propose a more general framework that exploits the fact that many word and multi-word TEs share a similar structure and/or morphology.

Each of these chapters starts with an introduction to the specific problem addressed, followed by a state-of-the-art review, a description of the proposed solution, and an evaluation of the results[14].

All chapters end with a brief summary, but the conclusions are deferred to Chapter 7, which concludes this thesis with an in-depth review of all contributions and a series of follow-up problems to be addressed in future work.

[14] Chapter 6 deviates slightly from this organization because it has a section describing a newly proposed pattern language before the state-of-the art review.

Chapter Two

Document Alignment

2.1 Introduction

The problem of *document alignment* is to find bilingual pairs of parallel documents within a potentially large multilingual collection of documents. It is a particularly relevant problem today, given the copious amount of interesting multilingual content available on the web and the heterogeneity and noise that is inherent to that source. Automatic document alignment methods allow us to align documents from web domains with hundreds or thousands of webpages, which would be impractical to to by hand.

In this chapter I propose a coverage-based document alignment method, which has been recently evaluated in the bilingual document alignment shared task of the First Conference on Machine Translation (WMT16), where it achieved the best performance over 19 competing systems submitted by 11 research groups.

Our method finds pairs of documents that are *more parallel to each other than* they are if compared *to any other document in the given set*. This means that the resulting document pairs are not necessarily parallel. However, this also means that this method can be applied to situations where we are not interested only in parallel documents, but also in comparable ones.

2.1.1 Harvesting Parallel Corpora from the Web

Since the late nineties, there have been several attempts at fully automating the creation of parallel corpora from the web. Some examples include the STRAND system by Resnik [92, 93], the BITS system by Ma and Liberman [52, 67], Bitextor by Esplà-Gomis et al [35] and PaCo2 by San Vicente and Manterola [112].

Harvesting parallel corpora from web takes four high-level steps, depicted in Figure 2.1:

Step 1 – Identify websites with interesting multilingual content. This step is typically implemented by submitting queries to web search engines such as Google and Bing and analyzing the returned results.

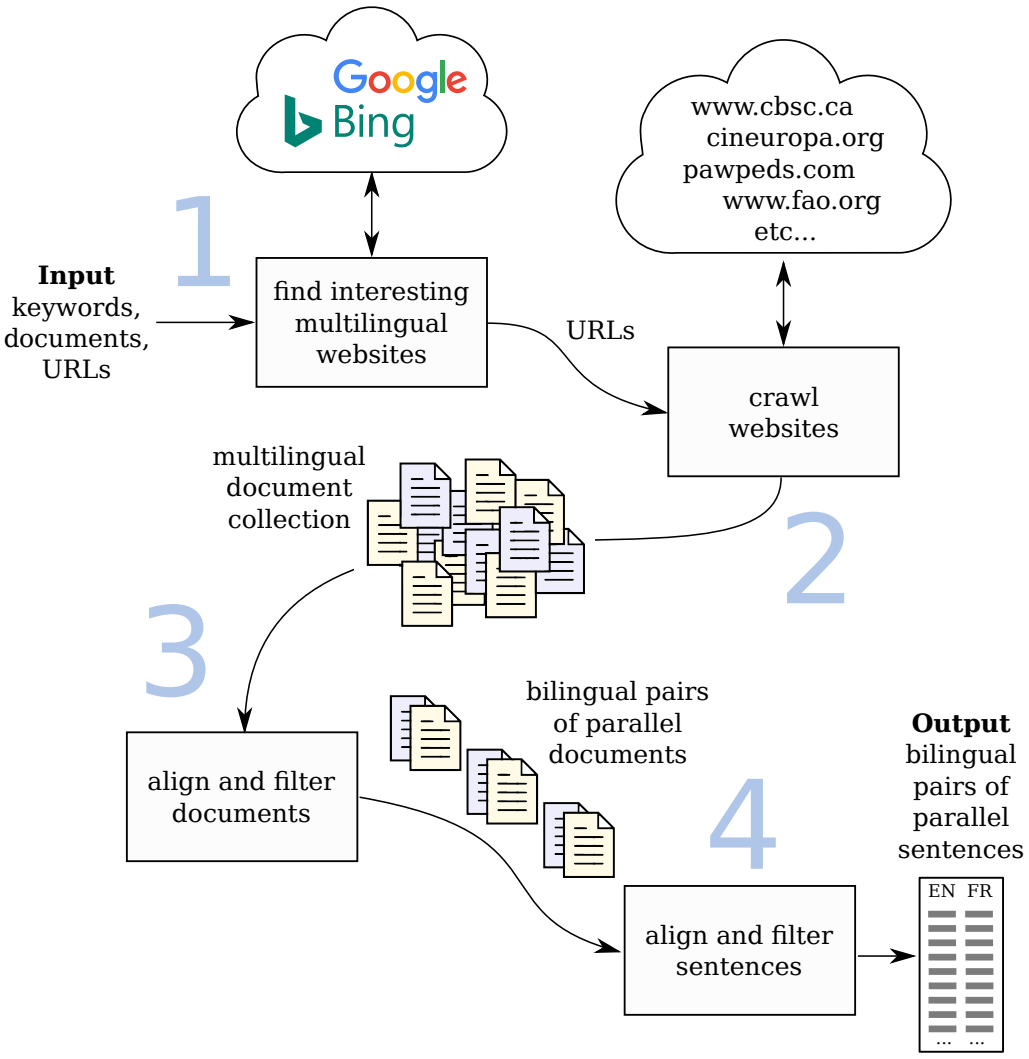


Figure 2.1: High-level steps for harvesting parallel texts from the web.

Step 2 – Crawl websites. This step is responsible for downloading textual content from websites and making it available locally in plain text format. This typically involves detecting the language and encoding of the web pages, removing boilerplate and extracting text from the HTML. The output of this step is a multilingual collection of plain text files.

Step 3 – Align downloaded documents. This is the subject of this chapter.

Step 4 – Align sentences within parallel documents. This is the subject of the next chapter.

Although interesting and relevant, I will not address the first two steps, as they diverge from the focus of this thesis.

2.1.2 Challenges

Of the three alignment problems addressed in this thesis (document-, sentence- and phrase-level alignments) the one addressed in this chapter is perhaps the easiest to solve. The main challenge is to deal efficiently with large collections of documents, because the number of possible pairings grows quadratically with respect to the collection size.

A naïve brute force approach would be to try to align at sentence level all possible bilingual pairs of documents from a given collection, and pick the ones that align best. This approach might be viable for collections of a few tens of documents. However, given the non-trivial computational cost of aligning documents at sentence level, this solution quickly becomes impractical as we increase the size of collections to hundreds or thousands of documents.

A scalable solution to the document alignment problem must either *prune the hypothesis space*, avoiding the quadratic growth by discarding most of the false hypotheses, or have a *fast test for parallelism*, i.e. much quicker than aligning two documents at sentence level.

2.1.3 General Algorithm

Most document alignment methods start with a candidate *generation* step followed by a *selection* or *filtering* step, or a combination of both, as depicted in Figure 2.2.

Not all alignment methods follow this algorithm, strictly. For example, instead of generating a few candidates for each document, Buck and Koehn [18] choose to test all possible pairings.

Candidate Generation

The candidate generation step is responsible for pairing hypothetically parallel documents, and assigning a score to each generated pair, reflecting the strength of the hypothesis. Each English document may be paired with several French documents ($1 : n$ candidate list), and vice versa ($m : 1$), as shown in the figure. Typically, the number of pairings in each direction is much smaller than the number of available documents in the "target" language, i.e. $m \ll M, n \ll N$.

Candidate Selection

The selection step is responsible for comparing candidates against each other and picking the best pair for each document (the one with highest score). Thus, the selection step does not warrant that the selected document pairs are indeed parallel, only that they are the best pairings possible for each document, according to the scoring function.

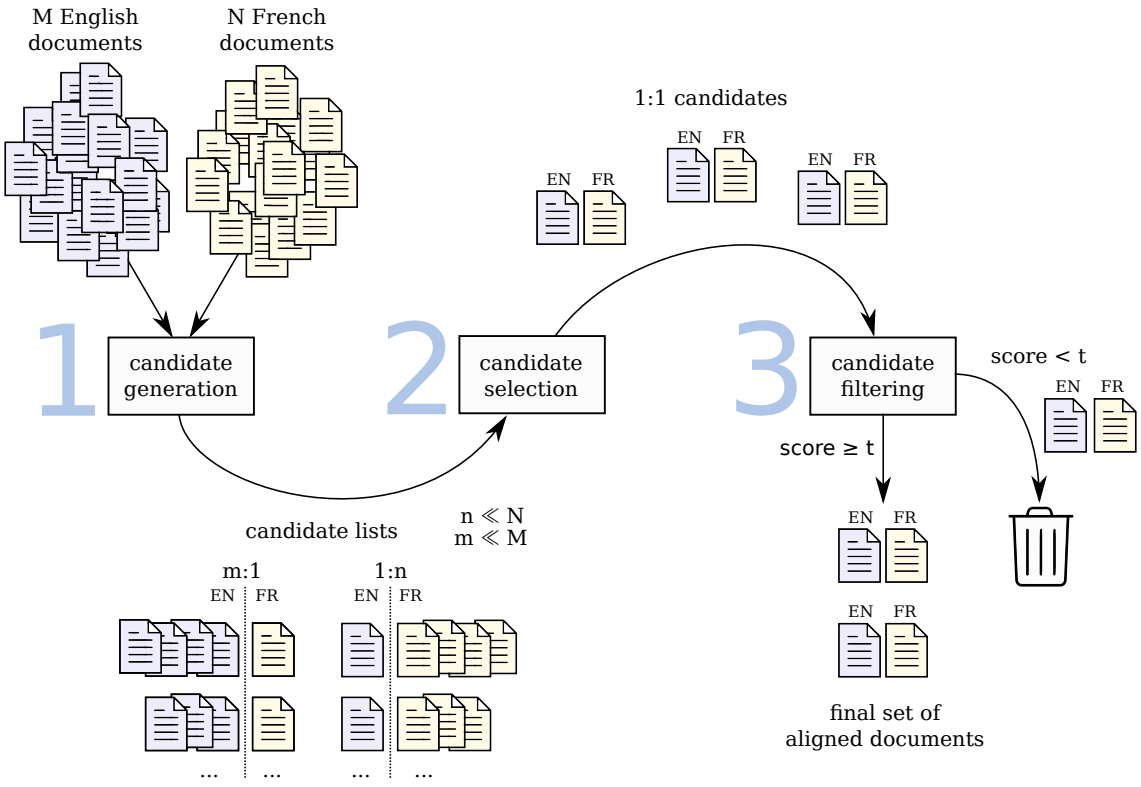


Figure 2.2: Typical document alignment algorithm.

Filtering

The filtering step aims at excluding non-parallel document pairs by keeping only those with scores higher than a given threshold value t. The threshold may be adjusted manually or based on a desired recall ratio using a development set of golden alignments.

2.2 State of the Art

Given that most alignment methods loosely follow the algorithm described above, we now turn our focus to the features used to score candidate pairs. The choice of features and how they are combined are the most distinguishing factors between alignment methods.

2.2.1 URL-matching

Most document alignment methods employ some kind of URL-matching technique, either for quick reduction of the hypothesis space, or as a similarity feature to be combined with content-based features. For example, the method proposed by Dara and Lin [29] starts by identifying parallel document pairs based on their URLs and those that could be matched are removed from further consideration, before moving on to a computationally costlier alignment algorithm based on document content. An opposite strategy is followed by

the Bitextor [35] alignment method, which employs a content-based method to generate a list of hypothetically parallel document pairs, and only then considers URL similarity, together with other features, to find the most parallel pairs among the generated candidates.

Since there is a large percentage of web domains where the URLs of parallel documents are parallel too, it is understandable that most alignment methods exploit this source of information. For example, let us consider the following groups of parallel URLs exhibiting different *types* of URL parallelism:

```
1 http://www.un.org/en/universal-declaration-human-rights/index.html
2 http://www.un.org/fr/universal-declaration-human-rights/index.html

3 http://eur-lex.europa.eu/homepage.html
4 http://eur-lex.europa.eu/homepage.html?locale=es
5 http://eur-lex.europa.eu/homepage.html?locale=fr

6 http://www.unl.pt/en/research/
7 http://www.unl.pt/pt/investigacao/
```

The URL-matching algorithm proposed by Smith et al [103] is targeted at the simplest *type* of URL-parallelism that we observe in the first two URLs. It starts by finding ISO-639 language identifier substrings[1] such as "en", "fr", "pt", "eng", "fra", "por", etc, in the URL, surrounded by any non-alphanumeric character (i.e. any character that is not a letter or a number). Then, any language identifiers found in a URL that correspond to the (auto-detected) language of the respective document are replaced by a generic placeholder string "*" (an asterisk) and compared against other (similarly modified) URLs. This algorithm is able to match URL 1 with 2 and 4 with 5 but none of the others.

In the second group of URLs (3–5) we have a slightly more complex type of parallelism where the URL of the English page (3) lacks the language-specifier component that is present in the corresponding Spanish and French pages (4 and 5). To handle these cases, Buck and Koehn [17] extended Smith et al's algorithm by collecting a list of language-specifier substrings, such as "locale=X", "lang=X", etc, to be removed before comparing URLs. This improved algorithm is able to match all the URLs in the second group (as well as the first).

The last group of URLs (6 and 7) has yet another type of parallelism, where some of the words within the URL have been translated, "research"↔"investigação" (note the missing diacritics in URL 7). This type of URL parallelism is not generally exploited, perhaps because of the increased ambiguity that arises from matching translated words,

[1] https://en.wikipedia.org/wiki/ISO_639

which is likely to produce false positives (i.e. matching URLs of pages which are not in fact parallel).

2.2.2 Textual Content Similarity

Many different ways of measuring content similarity of parallel documents have been proposed. There are broadly four strategies for comparing textual content across two different languages:

1. Apply machine translation to documents in one language and then compare the resulting translations with the documents in the other language. This strategy allows application of (monolingual) information retrieval techniques to rank documents according to their textual similarity [18, 29, 61].

2. Alternatively, cross-lingual information retrieval techniques such as cross-lingual latent semantic indexing[2] [11, 38] may be applied, avoiding the need to translate documents into a common language.

3. Use a supplied bilingual dictionary [35, 51, 52, 67, 100] or a statistical word translation model trained on seed sentence-aligned parallel corpora [7, 78] to match equivalent words in both languages.

4. Match tokens (or token sequences) that are common to both languages, such as proper nouns, numbers, URLs within pages, etc [51, 61, 65].

Each of these strategies, and particularly the ones based on information retrieval techniques, have too many variants and rely on complex background concepts that do not fit the scope of this thesis. However, as will be shown later in Section 2.3, it is possible to obtain similar or even better results with a (surprisingly) simple lexical coverage measure, which is a combination of 3 and 4.

2.2.3 Document Structure Similarity

In the late nineties, Resnik [92, 93] proposed an interesting method for comparing the structure of web pages, by first obtaining a linear abstract representation of the page structure, and then measuring the similarity of two pages by aligning their linear representations and measuring the number and size of matched and mismatched segments. An almost identical method, albeit slightly simpler, was later proposed by Esplà-Gomis et al [35], which is applicable to any XML document (as well as HTML, after conversion to XHTML) and works as follows:

First, a string representation of the document structure is obtained by replacing each different XML tag with a unique arbitrary character, and each sequence of N words between tags by a reserved word-representing character, repeated $\log_2(N)$ times, as shown in the example of Figure 2.3.

[2] https://en.wikipedia.org/wiki/Latent_semantic_indexing

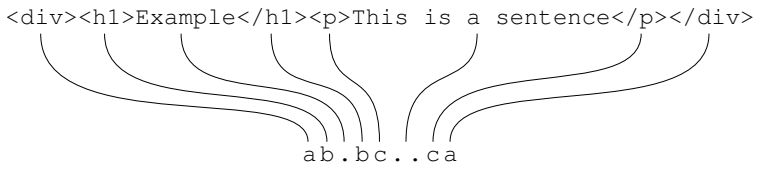


Figure 2.3: Document structure representation of a XHTML excerpt. Here, `a`, `b` and `c` represent the opening and closing `div`, `h1` and `p` tags respectively, and the dots represent the words within the tags. Note that the four words within the `p` tags are represented as two dots because $\log_2(4) = 2$, and the word within `h1` tags, despite $log_2(1) = 0$, is represented by one dot.

Finally, the structure of the two documents is compared by measuring the Levenshtein distance [62] between their string representations.

2.2.4 Feature Combination

Alignment methods combine the features described above in various ways. In the simplest case, features are employed in a cascading pipeline, where each feature is used in turn to align a subset of documents. For example, Dara and Lin [29] and Medved et al [78], first align documents based on URL matching, and then use a content-similarity score to align the remainder documents.

A different strategy, suggested by Germann [38] and Lohar et al [65], is to combine several features in a linear equation and then assign feature weights manually.

More sophisticated feature combination strategies are employed by Esplà-Gomis et al [35] and Papavassiliou [87], which employ regression models (a multilayer perceptron[3] and a support vector machine[4], respectively). The design choice of employing a regression model for combining the features has both its advantages and disadvantages. On one hand, this combination of features is more empirical and principled, but on the other, it requires a set of manually aligned documents for training the model. Furthermore, special care must be taken to ensure that the training set is representative of the type of documents to be aligned, under penalty of decreased model performance.

2.3 Proposed Coverage-Based Document Alignment Method

In this section I describe my proposed *coverage-based* document alignment method. First, I will present a coverage scoring function adequate to assess parallelism of a pair of documents, then we will see how to generate candidate document pairs and finally, how to select candidates based on coverage scores.

[3] https://en.wikipedia.org/wiki/Multilayer_perceptron

[4] https://en.wikipedia.org/wiki/Support_vector_machine

2.3.1 Measuring Coverage at Document Level

The general idea of coverage-based document alignment is to exploit the fact that parallel bilingual phrase pairs are more likely to co-occur in parallel documents than in non-parallel ones, especially for longer, more specific phrase pairs. Parallel phrase pairs, which include word pairs may occur by chance in non-parallel documents. However, the hypothesis that I put forward is that the number of parallel phrases within any given pair of parallel documents (E_i, F_j) should be higher than the number of parallel phrases that we would find in any other pairing of documents $(E_i, F_k), \forall k \neq j$ or $(E_l, F_j) \forall l \neq i$.

Coverage-based alignment does not assume that parallel documents will always have high coverage scores, because these will depend on the lexicon or phrase translation table. Instead we use coverage scores to compare alternative alignments and select the ones with highest scores, independently of the particular values of the score, which may be lower or higher depending on if we are using a small or large lexicon or phrase translation table.

2.3.2 Coverage Score

Let (e, f) denote a bilingual phrase pair, where e is an English phrase and f is a French phrase, without loss of generality when applying this method to other language pairs. Let (E, F) denote a bilingual pair of documents, where E is an English document and F is a French one. We say that a bilingual phrase pair (e, f) appears in a bilingual pair of documents (E, F), if e occurs in E and f occurs in F. We assign to each unique bilingual phrase pair an integer identifier tid. Associated with each tid we define a *binary feature* that takes value 1 if tid occurs in a document or 0 if it does not. Each English phrase e will be associated with one tid for each translation of that phrase. Likewise, each French phrase f will be associated with a tid for each translation of that phrase.

Thus, when an English phrase e occurs at least once in a given English document E, all binary features corresponding to $tids$ associated with e will have value 1 in E and all other features will have value 0. Analogously, when a French phrase f occurs at least once in a given French document F, all binary features corresponding to $tids$ associated with f will have value 1 in E and all other features will have value 0.

The number of distinct bilingual phrase pairs appearing in both documents is thus given by the cardinality of the intersection of the sets of features with value 1 in both documents.

Because each distinct bilingual phrase pair counts only as a single feature, independently of the length and the number of times that each of the phrases occurs in a document, we may say that we employ uniform weighting for all bilingual phrase pairs. In future work, non-uniform weighting could be investigated, as for example TF-IDF weigthing employed by Buck and Koehn [18].

Let get_doc_tids(D) be a function that returns the set of *tids* associated with phrases of document D. In Sub-Section 2.3.5 we will see how this function is implemented. We

define the *coverage ratio* of an English document E when paired with a candidate parallel French document F as the ratio of of the number of $tids$ associated with phrases of E that are also associated with phrases of F, as shown in Equation 2.1:

$$\text{fratio}(E,F) = \frac{|\text{get\_doc\_tids}(E) \cap \text{get\_doc\_tids}(F)|}{|\text{get\_doc\_tids}(E)|} \tag{2.1}$$

Here, we use operator $|\,|$ to denote the cardinality of a set and $\cap$ to denote intersection of two sets, as usual. Similarly, to compute the coverage ratio of a French document F when paired with a candidate English document E we simply swap the arguments in Equation 2.1, which will change the denominator to $|\text{get\_doc\_tids}(F)|$.

The coverage score of a candidate pair of documents, is given by a non-parametric combination of the two coverage ratios ($\text{fratio}(E,F)$ and $\text{fratio}(F,E)$). We prefer the geometric mean (Equation 2.2b) instead of the arithmetic (Equation 2.2a) or harmonic (Equation 2.2c) means, because it sits in the middle ground between the other two in terms of response to unbalanced inputs (see Equation 2.2d). In fact, the equalities between the three means (Equation 2.2d) only hold if the inputs a and b have the same value.

$$\text{AM}(a,b) = \frac{a+b}{2} \tag{2.2a}$$

$$\text{GM}(a,b) = \sqrt{ab} \tag{2.2b}$$

$$\text{HM}(a,b) = \frac{2ab}{a+b} \tag{2.2c}$$

$$\text{HM}(a,b) \leq \text{GM}(a,b) \leq \text{AM}(a,b) \tag{2.2d}$$

To better understand the choice of the geometric mean, let us consider for example three pairs of coverage ratios for three hypothetical pairings of documents: $(0.9, 0.1)$, $(0.65, 0.35)$ and $(0.5, 0.5)$. The arithmetic mean of each of these pairs is 0.5 (the same for all pairs) while the geometric mean is 0.3 for the first, 0.48 for the second and 0.5 for the third, which is the most balanced pair. Therefore, if we use the arithmetic mean, then we will not differentiate among these three cases, although the pair with more balanced coverage ratios is more likely to be parallel.

From observation we learned that extremely unbalanced coverage ratios typically indicate that one of the documents is much longer than the other. Since longer documents tend to have more unique phrases than shorter ones, whenever we compute the coverage ratios for such a pairing, the shorter document will have a greater coverage ratio than the the longer document. More precisely, the numerator of Equation 2.1 will be the same for $\text{fratio}(E,F)$ and $\text{fratio}(F,E)$, but the denominator will be larger for the document with more unique phrases.

The harmonic mean is slightly more sensitive to unbalanced input values than the geometric mean, and for the three pairings in the previous example we would get 0.18, 0.46 and 0.5, which are not too far from the respective geometric means.

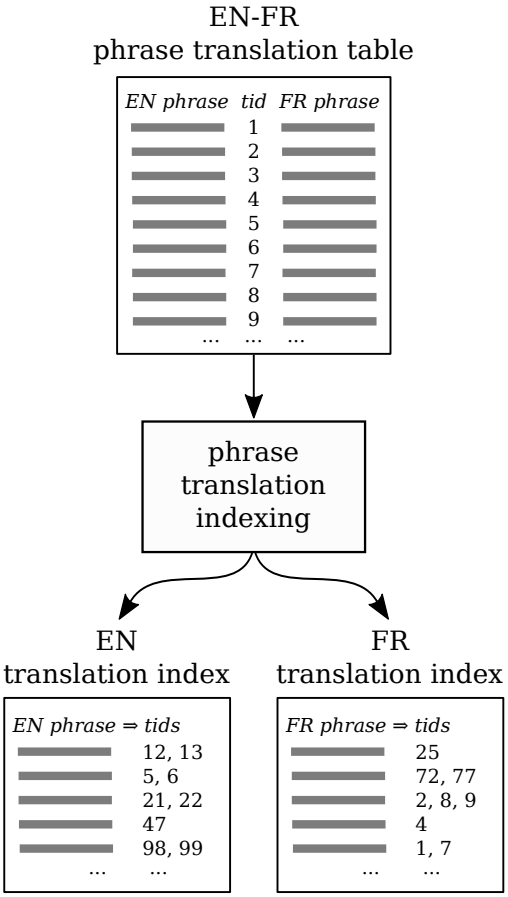


Figure 2.4: Phrase translation indexing.

In future experiments, one could replace the geometric with the harmonic mean and see how and if it affects performance of the whole alignment method.

Finally, replacing a and b in the geometric mean equation (2.2b) by the coverage ratio equations for both directions (2.1), we get the following equation for the coverage score:

$$\text{cscore}(E,F) = \sqrt{\text{fratio}(E,F)\,\text{fratio}(F,E)} \tag{2.3}$$

Each candidate pair of documents produced by the generation step, described ahead, is scored according to this equation.

2.3.3 Phrase Translation Indexing

Phrase translation indexing is a one-time pre-processing step that taks place before the proper alignment process begins.

This step, depicted in Figure 2.4, creates *phrase→tids* indices for both languages. Each line of the phrase translation table contains a unique phrase translation pair, but each of the two phrases may appear in multiple lines, since the same English phrase may have multiple French translations and vice versa. We use the line number of each phrase translation in the table as its tid. The indices constructed in this pre-processing step

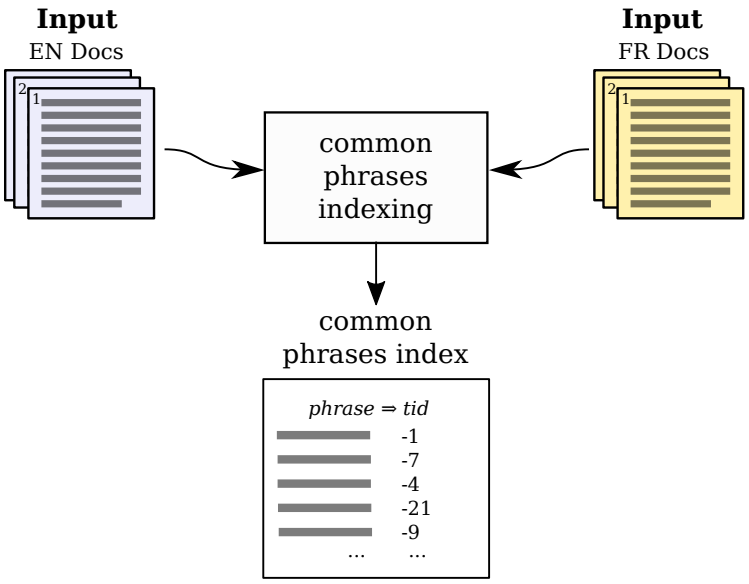


Figure 2.5: Common phrases indexing.

are hash tables mapping character strings (phrases) to lists of integers (tids) and are constructed as follows: we iterate over the phrase translation table, line by line, and insert each phrase in the hash table of the respective language, if not there already, and then we append the current line number to the list of *tids* associated with each phrase.

In Figure 2.4, the *tids* associated with English phrases are sequential, while in the *tids* associated with French ones are not. This is merely a consequence of the phrase translation table being sorted lexicographically by the English phrases and has no influence on the algorithm whatsoever.

2.3.4 Common Phrases Indexing

Common phrases indexing, depicted in Figure 2.5, is the first step when we start to align a document collection. It creates a *phrase→tid* index for phrases that occur in documents of both languages, which we call *common phrases*. Examples of common phrases are proper nouns, homographs, numbers and web addresses.

While in the phrase translation indices described above, a phrase may be associated with more than one *tid*, one for each translation of the phrase, here each phrase is associated with a single *tid* because we are assuming a phrase to be equivalent to itself. These indices are implemented as hash tables mapping character strings (phrases) to integers (tids).

Common phrase *tids* are negative integers to avoid clashes with *tids* from the phrase translation table.

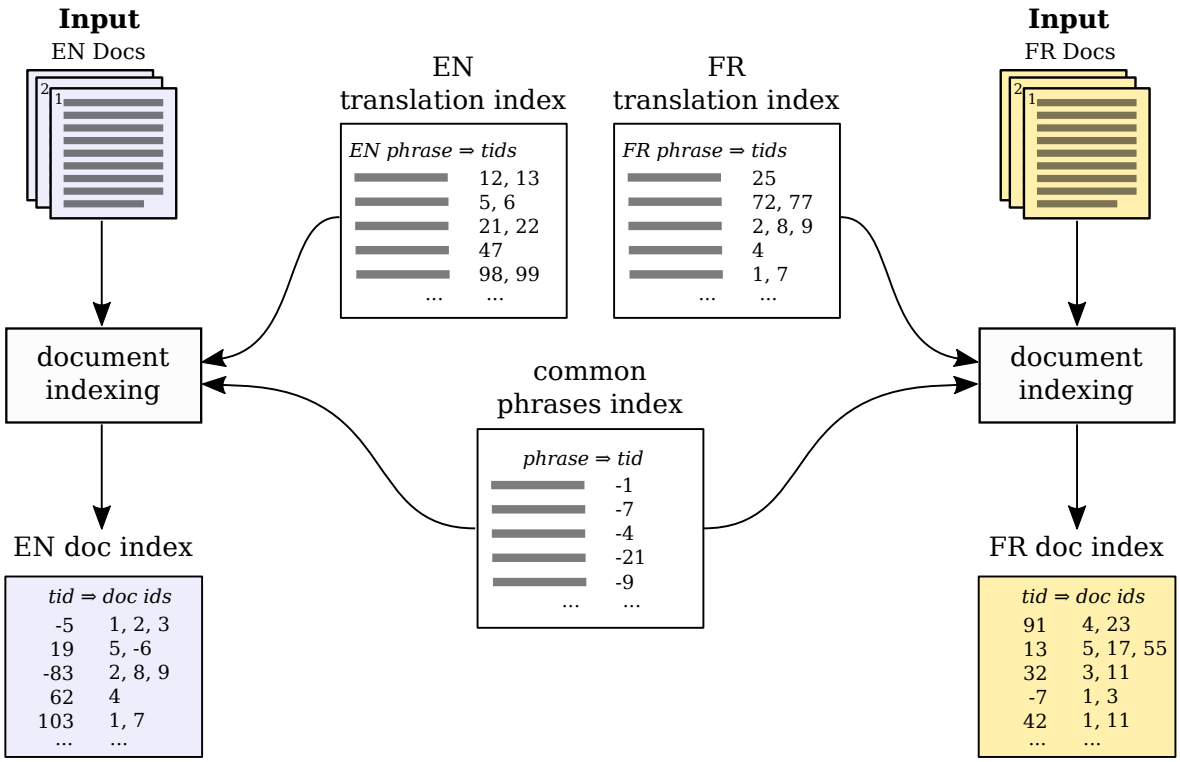


Figure 2.6: Document indexing.

2.3.5 Document Indexing

The document indexing step, depicted in Figure 2.6, is the second step executed when a new collection of documents is to be aligned, after the common phrases indexing. This step creates *tid→document ids* indices for both languages, which will be used in the generation step described in the next subsection. These indices are implemented as hash tables, mapping integers (*tids*) into lists of integers (document ids) and are created as follows: we iterate over the input documents and for each document we elicit all unique phrases with length 1 up to *max_phrase_len* tokens occurring in that document. Then, for each unique phrase we consult the phrase translation index of the corresponding language to get a list of *tids* associated with that phrase, which may be an empty list if the phrase is not in the index. We also consult the common phrases index and retrieve the *tid* associated with each phrase, if any.

We should set parameter *max_phrase_len* to the maximum length of phrases in the phrase translation table being used. Phrases in Moses phrase translation tables typically have a maximum length of 5 to 7 tokens.

Finally, we iterate over the *tids* collected for a given document, and we insert each *tid* into the *tid→documents* hash table of the respective language, if not there yet, and we append the id of the document to the list of document ids associated with that *tid*.

The time needed for creation of these document indices is proportional to the size of

the document collection.

2.3.6 Candidate Generation

The candidate generation step is responsible for balancing the time required to align a collection of documents with the precision and recall of the resulting alignments. If it generates too many candidates, then the process will take a long time to evaluate all generated candidates. If it generates too few candidates then there is an increased chance that some true parallel pairs are not among the generated candidates, and thus absent from the final output.

For the smaller web domains, we may generate all possible pairings, thus ensuring that all true parallel pairs are passed into the selection step. However, in the general case, we need to prune the hypothesis space and generate only a subset of all possible pairs.

We propose an heuristic for candidate generation that is applied in two directions, from English to French and in the opposite direction. In each run it generates a small number of target-language candidate pairs for each source-language document.

The number of candidate pairs generated for each document is not fixed. Instead, the method takes a parameter *min_cands* that specifies the minimum number of candidates to be generated for each document, if possible. As we will see further ahead, this parameter affects mostly the execution time of the algorithm and it does not require fine tuning. Also, note that this parameter specifies the minimum number of target-language candidates to be generated for each source language document, but not the maximum. As explained below, the actual number of candidates generated will normally be greater or equal than *min_cands*, but not much greater.

The candidate generation heuristic starts by invoquing function get_doc_tids to get a set of $tids$ associated with phrases in that document.

Then, *tids* are sorted by increasing frequency in terms of the number of target documents associated with each *tid*, by making use of the target-language document index. Thus, the first *tid* in the sorted list will typically be the identifier of a phrase translation that occurs only in one or two target documents, while the last *tid* will be the identifier of a phrase translation that occurs in almost all target documents.

Let trg_docs be an empty set of target-language documents. Next, we iterate over the frequency-sorted *tids* and for each *tid* we retrieve the list of target documents associated with it in the target-language document index, and we add those documents to trg_docs. As soon as trg_docs contains at least *min_cands* elements, we stop the iteration.

The final step consists of generating candidate pairs by pairing the source-language document with each target-language document in trg_docs.

This candidate generation heuristic method is executed from English to French and in the opposite direction. If we only generated candidates for one direction, say English-French, it could happen that the same French document was in the candidate lists of many English documents while other French documents were in none. By running the

Source	Source description	Minimum candidates	Time (sec)	Memory (MB)	Recall (%)
S1	phrase table + identical	25	1226	5859	91.75
		50	1492	5896	91.81
		100	1966	5934	91.87
		200	2786	5993	92.06
		400	4377	6121	92.12
S2	lexicon + identical	25	1056	5061	90.64
		50	1231	4636	90.76
		100	1621	5122	90.89
		200	2411	4744	91.07
		400	4113	5319	91.32
S3	phrase table	25	959	5035	91.56
		50	1074	5088	91.81
		100	1351	5127	91.75
		200	1814	5195	92.06
		400	2554	5431	92.18

Table 2.1: Alignment execution time, memory and recall measured over the development dataset using different sources of coverage information (S1-S3) and various values of min_cands.

algorithm in the reverse direction, we give the opportunity to every French document to also pick at least *min_cands* English documents as candidates.

After the two directional runs, we compute the union of the two generated sets of candidates, and that becomes our final list of candidate pairs which will be fed to candidate selection step, explained further ahead.

2.3.6.1 Experimentally Setting min_cands

As described above, parameter *min_cands* specifies the minimum number of candidate pairs generated for each document. Experimentally, we found that this parameter has little influence on the recall of the alignments produced by the method, but, as expected, it affects the execution time.

See the graphics in Figure 2.7 and Table 2.1 for a comparison of execution times, memory consumption and recall values, using the development dataset of the WMT16 bilingual document alignment shared task and considering a minimum of 25, 50, 100, 200 and 400 alignment candidates for each document. S1 to S3 correspond to different sources of coverage information, described ahead.

The development dataset used in these experiments was made available to the WMT16 shared task participants two months before the evaluation campaign took place, to let them develop and tune their alignment methods. It consists of a set of 49 document collections, each downloaded from a different webdomain, and totaling about 8.7GB of

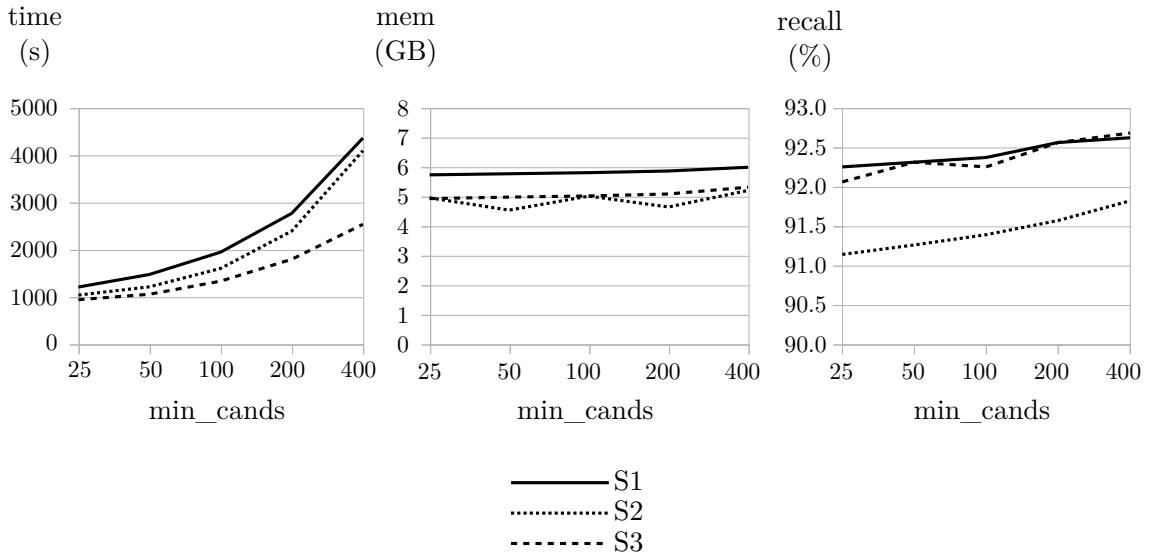


Figure 2.7: Alignment execution time, memory and recall measured over the developmentg dataset for various values of min_cands.

compressed HTML and text data. The dataset includes 1624 reference alignments, which were used to evaluate recall of the alignments produced.

From the first two plots we see that the execution time is more affected by *min_cands* than memory. As expected, a greater value of *min_cands* causes the alignment method to take longer. Note, however, that the number of minimum candidates is doubled at each datapoint collected, which means that the x axis is in a logarithmic scale (of base 2). Taking this into consideration, the variation of time with respect to *min_cands* is almost linear, instead of quadratic as suggested by the leftmost plot.

The document indexing step takes a substantial fraction of the total execution time and this step is not affected by the value of the *min_cands* parameter but instead depends mostly on the size of the document collection and the size of the phrase indices.

The S1, S2 and S3 plots in Figure 2.7 and the corresponding values in Table 2.1 correspond to experiments using three sources of coverage information:

S1 uses an English-French phrase translation table[5] obtained with Moses [55] from the Europarl corpus [53] and identical phrases to compute coverage.

S2 uses the ISTRION English-French bilingual lexicon and identical phrases to compute coverage.

S3 uses the same phrase table used by S1 but does not consider identical phrases.

Because there is a gain in recall of less than 1% from *min_cands*=25 to 400, but the execution time quadruplicates between these two settings, we chose *min_cands*=100 as a compromise between time and alignment quality. The 4377 seconds (1H13m) taken when *min_cands*=400 is perhaps not a long time, considering that the dataset has a size

[5]This table is available for download as part of the Moses 3.0 release from `http://www.statmt.org/moses/RELEASE-3.0/`

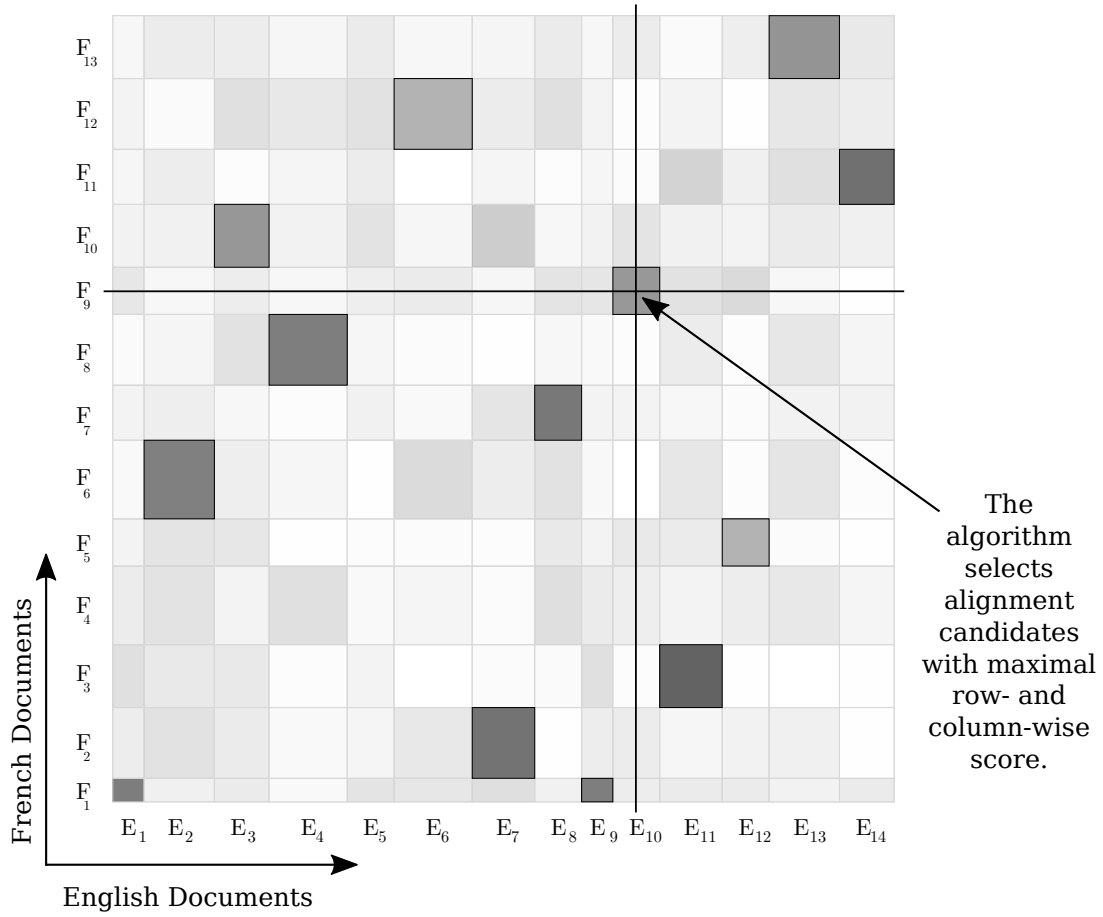


Figure 2.8: Document alignment hypothesis space

of 8.7GB (gzip-compressed).

2.3.7 Candidate Selection

The candidate selection step is responsible for selecting, among each group of *competing candidate pairs* (alternative hypotheses), the one with the maximum coverage score. The document alignment hypothesis space is depicted in Figure 2.8, where each small gray rectangle represents a single candidate pair. Instead of the full matrix of candidates shown here, the candidate generation algorithm will typically generate a sparse matrix of candidates with only a small fraction of all candidates possible. In this example, the full matrix was generated because the total number of documents of each language is lower than *min_cands*. The shade of gray of each rectangle indicates the coverage score of the respective candidate. Darker shades indicate higher scores.

We say that any two candidates are *competing candidates* if they are in the same row or column in the two-dimensional representation of Figure 2.8. We assume that only one pair of all competing candidate pairs is indeed parallel, i.e. there is at most one parallel French document for each English document and vice versa. Note that this assumption

will be correct only if the initial collection of documents has no exact duplicates[6]. Near duplicate documents pose no problem to the algorithm, since it will chose whichever version of those near-duplicate documents is the most parallel with a given document in the other language.

More formally, the selection algorithm selects pairs of documents (E_i, F_j) that verify the following two inequalities:

$$\text{cscore}(E_i, F_j) > \text{cscore}(E_i, F_k) \quad \forall k \neq j \tag{2.4a}$$

$$\text{cscore}(E_i, F_j) > \text{cscore}(E_l, F_j) \quad \forall l \neq i \tag{2.4b}$$

We call the selected (E_i, F_j) pairs as *maximal pairs*.

Note that the maximum candidate in a row does not always coincide with a maximum in its column. In other words, it may happen that a given English document E_i "prefers" a French document F_j that, in turn, "prefers" another English document E_k. To accommodate these situations, the selection algorithm is iterative and at each iteration, some maximal pairs are identified and removed from further consideration. Thus, if the pair (E_k, F_j) is maximal, it will be selected and both documents are removed from the hypothesis space. In the next iteration, since F_j is no longer available, document E_i will have to select the next French document with highest coverage in its row. The algorithm stops after an iteration where no maximal pairs are found.

2.4 Evaluating Document Alignment

This section presents the results of a large scale evaluation campaign of several alignment methods, including the proposed coverage-based method.

This evaluation was carried out in the context of the WMT16 *bilingual document alignment shared task* [17], where 11 teams (see Table 2.3) participated with 19 competing systems.

2.4.1 Evaluation Procedure and Datasets

The evaluation in the WMT16 bilingual document alignment shared task was based on recall, i.e. the ratio of URL pairs from the provided test set that were correctly identified by each evaluated alignment method. A one-to-one rule is enforced, which allows each English URL to be aligned with at most one French URL and vice versa.

The dataset used for the final evaluation campaign was a completely different dataset from the development dataset made available to the shared task participants two months in advance, and the reference alignments used for evaluation were not disclosed until after the evaluation campaign was finished. The evaluation itself was carried out by the

[6]Finding and removing exact duplicates in a document collection is easy to do. Thus, we assume it is done before alignment.

Method	Recall	# Predicted Pairs
NOVALINCS URL/COVERAGE	91.87%	148278
NOVALINCS COVERAGE/URL	90.53%	147857
NOVALINCS COVERAGE	72.78%	63207
BASELINE	67.92%	119979

Table 2.2: Evaluation results on the WMT16 development set.

shared task organizers. Each participant submitted a set of alignments and the shared task organizers compared these alignments with the reference alignments.

Full details about the content and preparation of the development and evaluation datasets are given in the paper describing the shared task [17].

Despite the merits of a pure content-based approach, which is applicable in scenarios where URLs and other metadata are not available, for this particular shared task we believed that we would obtain better results if we would take advantage of all information available (page URL and HTML structure) besides the plain text content.

Therefore, besides evaluating the coverage-based method on its own, we submitted two additional *hybrid* sets of results obtained by trivial combinations of the coverage-based method with the baseline URL-based method developed by Buck and Koehn [18].

The first extended set, called *NOVALINCS COVERAGE/URL*, gives priority to predictions of the coverage-based method, adding only URL-predicted pairs for URLs that were not aligned by the coverage-based method. Conversely, the second extended set, called *NOVALINCS URL/COVERAGE*, gives priority to the predictions of the URL-based method.

The English-French phrase translation table used by our coverage-based method was obtained from the Europarl corpus [53] and is part of the Moses 3.0 release[7].

2.4.2 Experimental Results

The results obtained with the coverage-based method and the two trivial combinations with the baseline method for the development set are summarized in Table 2.2. The final results obtained on the test set for all participating systems are given in Table 2.4.

The coverage-based alignment method improves 5% over the baseline on the development set and 26% on the test set. When combined with the baseline URL-based method, the recall is boosted up to 24% above the baseline on the development set and up to 35% on the test set. An explanation for the boosted recall is that since the two methods (coverage-based and URL-based) rely on completely different properties of the documents, their predictions are also, to some degree, complementary.

It is interesting that the coverage-based method made substantially fewer predictions than the baseline (about half) in the development set, and still yielded higher recall

[7] http://www.statmt.org/moses/RELEASE-3.0/

Acronym	Participant
ADAPT	ADAPT Research Center, Ireland [65]
BADLUC	University of Montréal, Canada [51]
DOCAL	Vicomtech [7]
ILSP/ARC	Athena Research and Innovation Center, Greece [87]
JIS	JIS College of Engineering, Kalyani, India [68]
MEDVED	Lexical Computing / Masaryk University, Slovakia [78]
NOVALINCS	Universidade Nova de Lisboa, Portugal [41]
UA PROMPSIT	University of Alicante / Prompsit, Spain [34]
UEDIN COSINE	University of Edinburgh, Scotland — Buck [18]
UEDIN LSI	University of Edinburgh, Scotland — Germann [38]
UFAL	Charles University in Prague, Czech Republic [61]
YSDA	Yandex School of Data Analysis, Russia [100]
YODA	Carnegie Mellon University [29]

Table 2.3: List of participants in the WMT16 Bilingual Document Alignment Shared Task.

(+4.86%). The same is verified when we compare the top-performing system, our URL/-COVERAGE combination, with the second and third best systems, YODA and UEDIN1 COSINE: the URL/COVERAGE system made 235,812 predictions, while YODA made 318,568 and UEDIN1 COSINE made 368,260. These figures support some speculation that the precision of the coverage-based alignment may be higher than the precision of the other systems, because of the 1-1 restriction imposed in the evaluation. Since each English document can only be aligned with one French document and vice versa, each incorrectly aligned pair of documents (E_i, F_j), will be penalized twice as much in the recall: neither E_i can be aligned with another French document nor F_j can be aligned with another English document.

The two best performing systems, URL/COVERAGE and YODA, are relatively similar: both combine URL-matching with a content-based similarity measure and both give priority to the alignments made by URL matching. However, while YODA requires machine translation of the French documents to English before alignment, the coverage-based method does not.

The third best system, UEDIN1 COSINE, has the merit of being the best performing system that is completely content-based, i.e. it does not take advantage of URL matching. However, like YODA, it has the disadvantage of requiring French texts to be machine translated.

In future evaluations it would be interesting to measure alignment precision along with recall.

2.5 Summary

In this chapter I addressed the problem of bilingual document alignment. The proposed coverage-based method has several interesting properties:

Name	Predicted Pairs	Pairs After 1–1 Rule	Found Pairs	Recall %
NOVALINCS URL/COVERAGE	235 812	235 812	2 281	95.0
YODA	318 568	318 568	2 256	93.9
UEDIN1 COSINE	368 260	368 260	2 140	89.1
NOVALINCS COVERAGE/URL	235 763	235 763	2 129	88.6
DOCAL	191 993	191 993	2 128	88.6
UEDIN2 LSI-V2	367 948	367 948	2 105	87.6
NOVALINCS COVERAGE	207 022	207 022	2 060	85.8
UEDIN2 LSI	681 744	271 626	2 062	85.8
ILSP-ARC-PV42	291 749	287 860	2 040	84.9
UFAL-4	1 080 962	268 105	2 023	84.2
YSDA	277 896	277 896	2 021	84.1
UA PROMPSIT BITEXTOR 5.0	157 682	157 682	2 001	83.3
UFAL-1	592 337	248 344	1 953	81.3
UFAL-3	574 434	207 358	1 938	80.7
MEDVED	155 891	155 891	1 907	79.4
BADLUC	681 610	263 133	1 905	79.3
UFAL-2	574 433	178 038	1 901	79.1
UA PROMPSIT BITEXTOR 4.1	95 760	95 760	748	31.1
ADAPT	61 094	61 094	644	26.8
ADAPT-V2	69 518	69 518	651	27.1
JIS	323 929	28 903	48	2.0
URL (baseline)	148 537	148 537	1 436	59.8

Table 2.4: Official Results of the WMT16 Bilingual Document Alignment Shared Task.

1. it is language and domain independent;
2. it is able to align documents with varying degrees of parallelism, ranging from barely comparable documents to fully parallel ones;
3. it is content-based, which makes it applicable in a wider range of scenarios than other approaches relying too much on URLs or document structure;
4. unlike other content-based approaches, the coverage-based method does not need to translate the documents into a common language in order to align them;
5. it takes advantage of existing knowledge encoded in phrase translation tables.

When combined with the URL-matching method, the coverage-based method achieved the top performance in the WMT16 Bilingual Document Alignment Shared Task. Despite the good performance of this method in this evaluation campaign, there are at least two aspects that could be improved in future work:

Feature combination URL-matching and the coverage score should be combined in a more principled manner using a regression model as proposed by Esplá-Gomis et al [35] and Papavassiliou [87].

Feature weighting In the same way that Buck and Koehn [18] used TF-IDF to weight n-grams, phrase translations from the phrase translation table could be weighted with TF-IDF or a similar score.

Chapter Three

Sentence Alignment

3.1 Introduction

Sentence alignment is the task of identifying parallel sentences within parallel documents. If two documents are perfectly parallel, then each sentence of one document has a parallel counterpart in the other, and they follow exactly the same order in both documents. In this ideal situation, we would align sentences from both documents pairwise, the first of one document with the first of the other, the second with the second and so on and so forth.

In the real world, however, documents are seldom perfectly parallel, because sentences may be merged, split, inserted or deleted in translation. Additionally, given the floating nature of figures, tables, headers, footers and the like, these often appear in non-parallel positions of the documents, intermixed with normal running text. Glossaries are another cause of large non-parallel sections within documents. Because glossary entries appear lexicographically sorted, the glossaries will generally have different orderings in different languages, and the larger the glossaries, the larger these non-parallel sections will be.

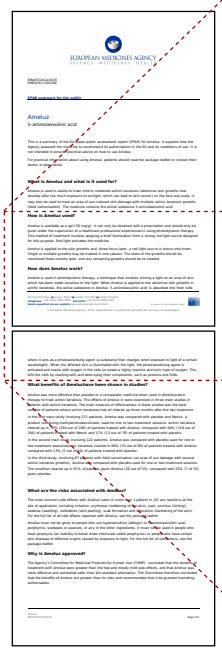
Texts extracted from PDF[1] or documents digitized via optical character recognition pose additional problems. While PDF text extractors have heuristics to deal with multi-column documents, they are rarely able to cope with tables or figures appearing side-by-side with running text. In these situations, the text at the side will often be mixed with content from the tables or figures, resulting in non-parallel and sometimes unintelligible sections.

For all these reasons, aligning documents at sentence level is a non-trivial problem.

Figure 3.1 shows a pair of parallel PDF documents, retrieved from the European Medicines Agency (EMA), where the page breaks of both documents occur in non-parallel positions with respect to the running text. As a consequence, when we extract and align the text from these PDFs, the text from the footers will appear intermixed with the main

[1] Portable Document Format is a widespread format for exchanging documents formatted for printing.

English Document



How is Ameluz used?

Ameluz is available as a gel (78 mg/g). It can only be obtained with a prescription and should only be given under the supervision of a healthcare professional experienced in using photodynamic therapy. This method of treatment involves applying a brief illumination from a strong red light source designed for this purpose. Red light activates the medicine.

Ameluz is applied to the skin growths and, three hours later, a red light source is shone onto them. Single or multiple growths may be treated in one session. The state of the growths should be monitored three months later, and any remaining growths should be re-treated.

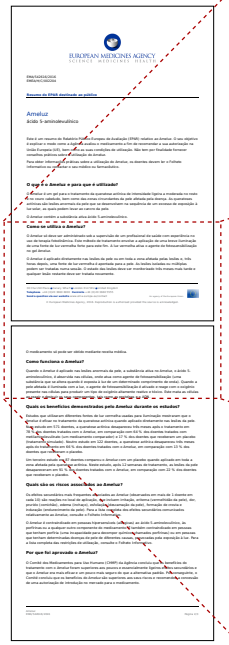
How does Ameluz work?

Ameluz is used in photodynamic therapy, a technique that involves shining a light on an area of skin which has been made sensitive to the light. When Ameluz is applied to the abnormal skin growths in actinic keratosis, the active substance in Ameluz, 5-aminolaevulinic acid, is absorbed into their cells

30 Churchill Place ● Canary Wharf ● London E14 5EU ● United Kingdom
Telephone +44 (0)20 3660 6000 **Facsimile** +44 (0)20 3660 5555
Send a question via our website www.ema.europa.eu/contact

where it acts as a photosensitising agent (a substance that changes when exposed to light of a certain wavelength). When the affected skin is illuminated with the light, the photosensitising agent is activated and reacts with oxygen in the cells to create a highly reactive and toxic type of oxygen. This kills the cells by reacting with and destroying their components, such as proteins and DNA.

Portuguese Document



Como se utiliza o Ameluz?

O Ameluz só deve ser administrado sob a supervisão de um profissional de saúde com experiência no uso de terapia fotodinâmica. Este método de tratamento envolve a aplicação de uma breve iluminação de uma fonte de luz vermelha forte para este fim. A luz vermelha ativa o agente de fotossensibilização no gel Ameluz.

O Ameluz é aplicado diretamente nas lesões da pele ou em toda a zona afetada pelas lesões e, três horas depois, uma fonte de luz vermelha é apontada para a pele. As lesões isoladas ou múltiplas podem ser tratadas numa sessão. O estado das lesões deve ser monitorizado três meses mais tarde e qualquer lesão restante deve ser tratada novamente.

30 Churchill Place ● Canary Wharf ● London E14 5EU ● United Kingdom
Telephone +44 (0)20 3660 6000 **Facsimile** +44 (0)20 3660 5555
Send a question via our website www.ema.europa.eu/contact

O medicamento só pode ser obtido mediante receita médica.

Como funciona o Ameluz?

Quando o Ameluz é aplicado nas lesões anormais da pele, a substância ativa no Ameluz, o ácido 5-aminolevulínico, é absorvida nas células, onde atua como agente de fotossensibilização (uma substância que se altera quando é exposta à luz de um determinado comprimento de onda). Quando a pele afetada é iluminada com a luz, o agente de fotossensibilização é ativado e reage com o oxigénio presente nas células para produzir um tipo de oxigénio altamente reativo e tóxico. Este mata as células ao reagir e destruir os seus componentes, tais como as proteínas e o ADN.

Figure 3.1: Example parallel PDF documents with non-parallel page breaks.

Id	English segment	Portuguese segment
1	The state of the growths should be monitored three months later, and any remaining growths should be re-treated.	O estado das lesões deve ser monitorizado três meses mais tarde e qualquer lesão restante deve ser tratada novamente.
2	How does Ameluz work?	
3	Ameluz is used in photodynamic therapy, a technique that involves shining a light on an area of skin which has been made sensitive to the light.	
4	When Ameluz is applied to the abnormal skin growths in actinic keratosis, the active substance in Ameluz, 5-aminolaevulinic acid, is absorbed into their cells	
5	30 Churchill Place · Canary Wharf · London E14 5EU · United Kingdom	30 Churchill Place · Canary Wharf · London E14 5EU · United Kingdom
6	Telephone +44 (0)20 3660 6000 Facsimile +44 (0)20 3660 5555	Telephone +44 (0)20 3660 6000 Facsimile +44 (0)20 3660 5555
7	Send a question via our website www.ema.europa.eu/contact	Send a question via our website www.ema.europa.eu/contact
8	An agency of the European Union	An agency of the European Union
9	© European Medicines Agency, 2016.	© European Medicines Agency, 2016.
10	Reproduction is authorised provided the source is acknowledged.	Reproduction is authorised provided the source is acknowledged.
11		O medicamento só pode ser obtido mediante receita médica.
12		Como funciona o Ameluz?
13	where it acts as a photosensitising agent (a substance that changes when exposed to light of a certain wavelength).	Quando o Ameluz é aplicado nas lesões anormais da pele, a substância ativa no Ameluz, o ácido 5aminolevulínico, é absorvida nas células, onde atua como agente de fotossensibilização (uma substância que se altera quando é exposta à luz de um determinado comprimento de onda).
14	When the affected skin is illuminated with the light, the photosensitising agent is activated and reacts with oxygen in the cells to create a highly reactive and toxic type of oxygen.	Quando a pele afetada é iluminada com a luz, o agente de fotossensibilização é ativado e reage com o oxigénio presente nas células para produzir um tipo de oxigénio altamente reativo e tóxico.

Figure 3.2: Mis-alignment of texts extracted from PDFs shown in Figure 3.1 caused by footers. Segments 2 to 4, 11 and 12 are aligned with empty segments due to non-parallelism introduced by the footers. Segment 13 is only partially correct, since the English sentence begins on segment 4.

running text at non-parallel positions, resulting in incorrect alignments, as shown in Figure 3.2.

Because the footers in these documents are relatively large and identical in both languages, the alignment algorithm preferred to align the footers instead of the surrounding segments.

One might think that since the footers of these documents are exactly equal, we could apply an heuristic filter to discard segments having exactly the same text in both languages. However, this type of cleanup is highly dependent on the documents being considered and not always possible. While in these particular documents the footers are identical, in other documents they may be different.

Structured document formats, such as HTML, DOCX[2] and OpenDocument[3], typically allow much cleaner text extraction than from PDF, avoiding intermixing the aforementioned floating materials (figure captions, headers, etc.) with running text. Unfortunately, a large portion of parallel documents available on the web are in PDF format only, and thus we must cope with that format.

3.1.1 Importance of Sentence Alignment

Sentence alignment is a fundamental pre-processing task for corpora-based approaches to machine translation (MT), such as phrase-based statistical machine translation (PB-SMT) [56], hierarchical phrase-based statistical machine translation (HPBSMT) [19], neural machine translation (NMT) approaches [8], and hybrid approaches such as TectoMT [115]. All of these MT approaches require pairs of parallel sentences as input data.

Furthermore, sentence alignment is also useful for creating translation memories (TMs) for use in computer assisted translation (CAT) systems. Typically, TMs are created during the translation process by CAT tools and they are kept private to the individual or company that performed the translation job. TMs are a valuable asset in the competitive market of translation services because they provide a productivity boost if they are of high quality and their content is similar to the new texts to be translated. By sentence aligning translated documents we are able to (re-)create translation memories, thus enabling other translators to benefit from the productivity boost. Finally, another use of sentence alignments is the creation of bilingual concordances [26, 46], which allow linguists, lexicographers, translators and language learners to search individual expressions or bilingual pairs of expressions, and see their contextual (co-)occurrences in parallel corpora.

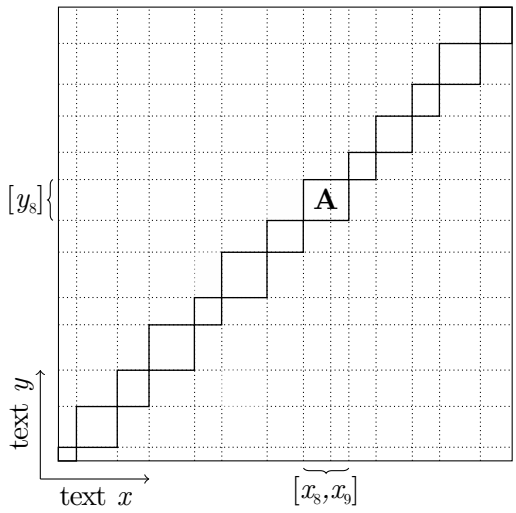


Figure 3.3: Sentence alignment hypothesis space. Texts are represented as the horizontal and vertical axes and dotted lines represent sentence boundaries. Each of the smallest rectangles, framed by dotted lines, represents a possible 1:1 alignment. True alignments are outlined with solid lines. Note the 2:1 alignment represented by rectangle **A**.

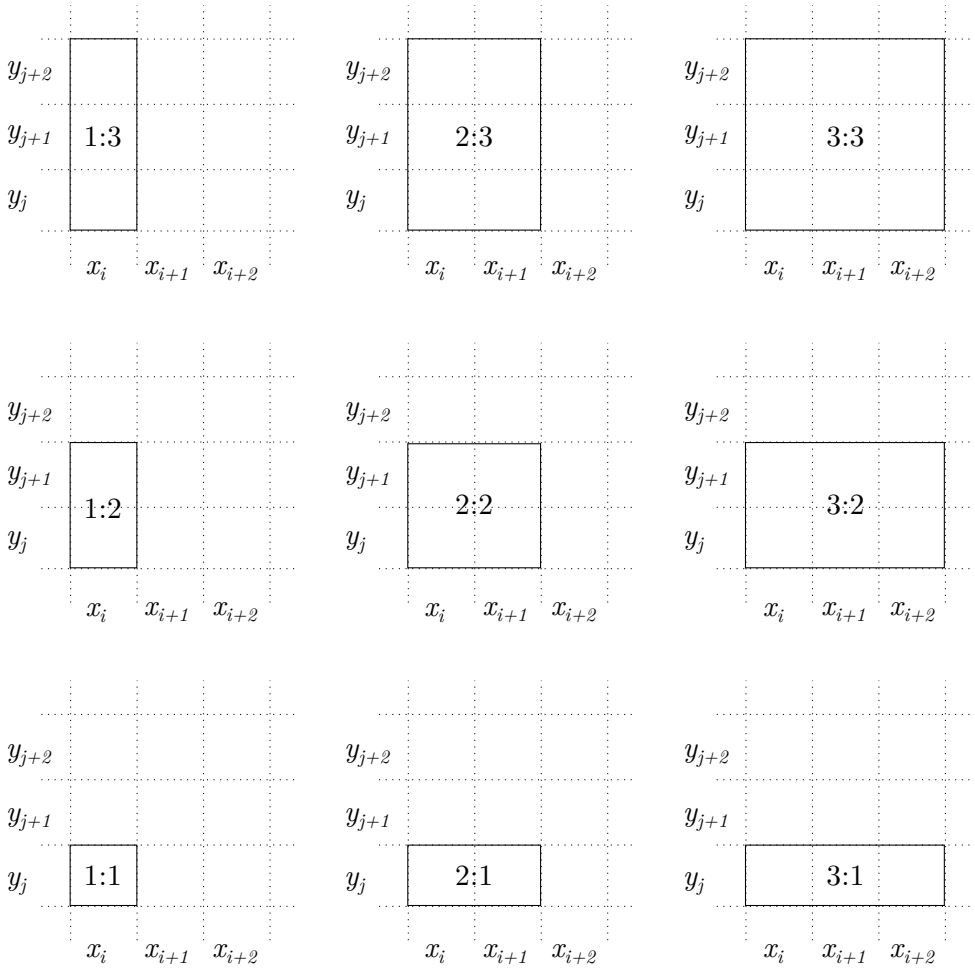


Figure 3.4: Most commonly considered sentence alignment configurations.

3.1.2 Geometry of the Sentence Alignment Problem

The sentence alignment problem has a two-dimensional geometric representation, shown in Figure 3.3. The x and y axes represent the two documents being aligned and each point along these axes corresponds to a character-based position in the respective document. The bottom left and top right corners of the xy plane correspond to the beginning and end of both documents, respectively. The horizontal and vertical dotted lines mark sentence boundaries in the texts[4] and each small rectangle framed by sentence boundaries represents a 1:1 alignment hypothesis, i.e. one sentence aligned with one sentence.

Besides 1:1 alignments, which are the most common, other alignment configurations are possible. For example, the rectangle **A** in Figure 3.3 represents a 2:1 alignment between the segments $[x_8, x_9]$ and $[y_9]$. The 2:1 and 1:2 configurations represent two sentences being translated as a single sentence and vice versa.

Sometimes sentences in one text do not have corresponding sentences in the other. To represent such situations we admit 0:1 and 1:0 alignment configurations in the final set of sentence alignments for a given pair of documents. Other alignment configurations with empty segments on one side, such as 2:0, 0:2, 3:0, 0:3, etc, are not necessary, since they can be equivalently represented by multiple consecutive 1:0 or 0:1 alignments. For example, segments 2, 3 and 4 of Figure 3.2 are of type 1:0, which are equivalent to a single segment of 3:0. Likewise, segments 11 and 12 are of type 0:1 and are equivalent to a single segment of type 0:2.

In theory, many other configurations are possible. In practice, however, situations where more than three sentences are translated together are very rare and doubtfully parallel or small enough to be useful for downstream applications. Figure 3.4 illustrates the six most common alignment configurations, except for 1:0 and 0:1, which are not represented because they have no area. Most alignment methods consider only these eight configurations or fewer.

The two-dimensional representation in Figure 3.3 makes it easier to understand the quadratic space and time complexity growth with respect to the length of the texts to be aligned. The area of the xy-plane representing the hypothesis space grows quadratically with respect to the length of the texts, as does the number of alignment hypotheses.

The true sentence alignments for this pair of texts are represented as rectangles outlined with continuous lines. Note how these rectangles line up from the bottom left corner of the hypothesis space, which corresponds to the beginning of both texts, to the top right corner, which corresponds to the end of both texts.

Hopefully, this example makes it easier to understand the monotonicity constraint

[2]The Office Open XML format, standardized as ECMA-376 and ISO/IEC 29500. https://en.wikipedia.org/wiki/Office_Open_XML

[3]The Open Document Format for Office Applications standardized by ISO/IEC 26300. https://en.wikipedia.org/wiki/OpenDocument

[4]The documents depicted in this figure are very short for clarity. Typically, documents will have hundreds or thousands of sentences.

employed by most aligners, which requires that aligned segments should be oriented strictly diagonally, from the beginning to the end of texts. This constraint is based on the assumption that corresponding sentences on both texts are not reordered, except for *close range reorderings* of two or three adjacent sentences which should be encapsulated in a single segment.

3.2 State of the Art

Initial work on sentence alignment, in the early nineties, apparently demonstrated that high-accuracy alignments[5] could be easily obtained with methods that considered only the length of sentences, completely disregarding their lexical content [15, 37]. These methods rely on the expected length proportionality between sentences and their translations. Short sentences are typically translated as short sentences and long sentences as long sentences. The high accuracies obtained in these early experiments are, unfortunately, not representative of the effectiveness of lenght-based methods in general. Instead, the reported high-accuracies are on the merits of the easy to align documents used in those experiments.

Bleualign is a state-of-the-art sentence alignment method with top-performance on noisy texts [2, 99]. The general idea of Bleualign is to translate one of the texts (with a MT system) and then align the translation with the other text using the sentence-wise BLEU score [88] as an indicator of sentence similarity.

Besides its good performance on noisy texts, Bleualign is interesting because it takes advantage of previously acquired translation knowledge that is encoded within the MT system being used to translate the texts to be aligned. By contrast, standalone alignment methods such as the Microsoft Bilingual Aligner [81], Hunalign [110] and Gargantua [14] automatically infer a word-based translation lexicon from the texts being aligned as they proceed. Consequently, the performance of these methods degrades when the texts to be aligned are short as there is less data to support statistical inference of a bilingual lexicon.

In our view, standalone alignment methods are more suited to scenarios where no parallel corpus exists for the language pair under consideration. But as more corpora become available, these scenarios are becoming less frequent.

Instead, the most common scenario today is perhaps one where we want to align new texts and add them to an existing corpus. In this case, our point is that the alignment method should take advantage of the existing corpus to align the new texts.

Some alignment methods, such as Champollion [63, 66] and Hunalign [110], are able to use bilingual word lexica, but there are no guidelines how to obtain these lexica, nor how the size and quality of the lexica relate to quality of alignments. We also observe that longer and less frequent phrases such as the German “die Europäische Zentralbank” and the French “la Banque Centrale Européenne” (the European Central Bank) tend to be

[5] With reported precisions ranging from 96% to 99%.

much more reliable indicators of sentence-parallelism than single words. Hence, instead of a word-to-word translation lexicon, the coverage-based sentence alignment method proposed in the next section is based on a phrase translation table.

Compared to the alignment method proposed in my MSc thesis [39], which relies on a partially validated bilingual phrase lexicon, the coverage-based method proposed in the next section has the advantage of being able to use phrase translation tables from Phrase-Based Statistical Machine Translation (PBSMT) systems such as Moses [55]. Since these phrase tables are obtained in an unsupervised manner for any language pair, the coverage-based alignment method is equally applicable to any language pair, even for those that we do not have a bilingual lexicon yet.

While Bleualign [99] is a sentence alignment algorithm explicitly designed to take advantage of previously acquired knowledge, although in an indirect way through a MT system, the scoring function that I will introduce next, makes direct use of knowledge encoded in Moses phrase tables [55]. Not only this solution is algorithmically simpler, if we consider the MT system as part of the Bleualign method, but is also more resource-efficient and effective, as we will see later in the evaluation section.

3.3 Proposed Coverage-Based Sentence Alignment Method

While Bleualign approaches the alignment problem as a monolingual similarity-maximization problem, I approach the problem as a coverage-maximization problem. As mentioned before, Bleualign requires translation of one of the texts into the language of the other because BLEU is only applicable to text segments of the same language. The *coverage* score that I propose as a replacement of BLEU, works with bilingual text segments, thus avoiding the need to translate. Like the coverage-based document alignment method proposed in the previous chapter, here we will compute coverage with respect to a phrase translation table obtained with the Moses toolkit [55]. The first step, described below, is to find all co-occurrences of phrase translations from the phrase translation table in each pair of segments considered. To that purpose, we will use the same phrase translation indices that we created for the coverage-based document alignment method. The creation of these indices was described in Subsection 2.3.3.

3.3.1 Matching Phrase Translations in Segments

The input to this step is a pair of segments, each one composed of one or more contiguous sentences. The output is a list of occurrences of phrase translations. An occurrence of a phrase translation is a pair of phrase occurrences, one in each input segment. In turn, a phrase occurrence is defined by a (*begin*, *end*) pair of character-based positions, which indicate where the phrase begins and ends within the segment.

In our usual two-dimensional representation, an occurrence of a phrase translation corresponds to a rectangle within the larger rectangle defined by the aligned segments.

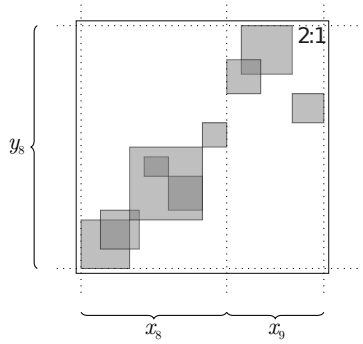


Figure 3.5: Matched phrase translations within a pair of parallel segments.

For example, Figure 3.5 illustrates a 2:1 alignment between segment $[x_8, x_9]$ and $[y_8]$ containing nine matched phrase translations.

Now that we know what are the inputs and outputs of this step, let us begin the procedure for matching phrase translations in parallel segments.

Let $occs_x$ be an empty map of *tid→occurrences* (integers to rectangles).

For each contiguous phrase with length 1 up to *max_phrase_len* within the x segment, we check if the phrase is in the *phrase→tids* index of the x language. If it is, then we retrieve the list of tids associated with the phrase and we add each tid to the $occs_x$ map, if not there already, and we append the character-based positions where the phrase begins and ends to the associated list of *occurrences*.

Like in the coverage-based document alignment method proposed in the previous chapter, we should set parameter *max_phrase_len* to the maximum length of phrases in the phrase translation table being used.

After iterating over all contiguous phrases of length 1 up to *max_phrase_len* within the x segment, we repeat all the steps from the beginning until this point for the y segment, thus producing $occs_y$.

Next, for each tid that is present in both $occs_x$ and $occs_y$, we will generate phrase translation occurrences by pairing each phrase occurrence in $occs_x[tid]$ with every phrase occurrence in $occs_y[tid]$. In other words, we are generating the cartesian product between phrase occurrences in segment x and phrase occurrences in segment y.

3.3.2 Coverage Score

After matching phrase translations in a bilingual pair of segments, coverage is measured as the ratio of text that is covered by those occurrences in both languages.

More specifically, we take the set of rectangles produced in the previous step, which we will refer as $\boldsymbol{T}$, and we project these rectangles into the y and x axes, as shown on the left- and right-hand sides of Figure 3.6 (c). The projections into the y and x axes are represented as light gray "shadows" to the left and below the phrase translations, respectively. In this example, segments $[y_8]$ and $[x_8, x_9]$ are both fully covered.

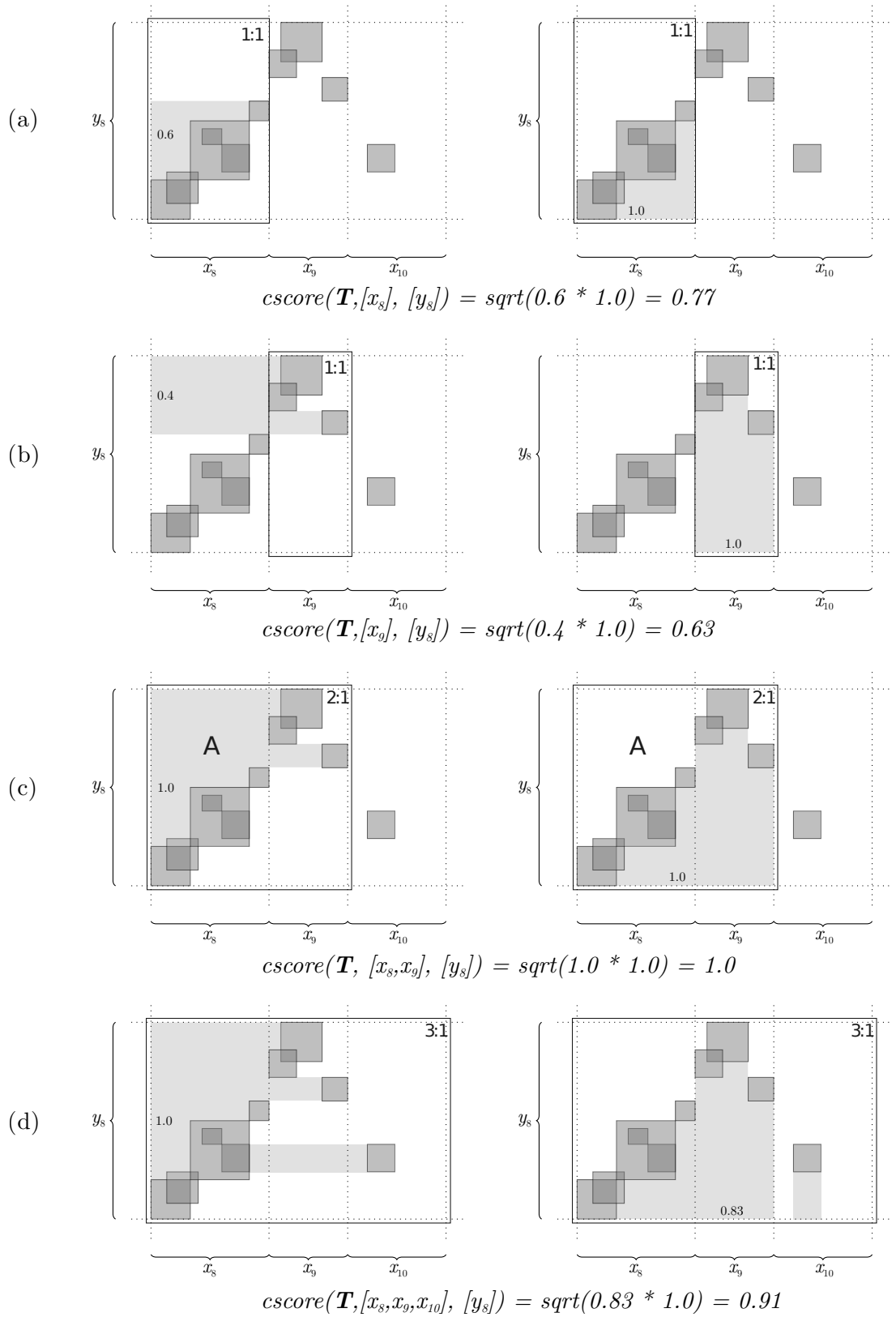


Figure 3.6: Detail of how coverage is calculated from matched phrase pairs within four alternative alignments: (a) and (b) are 1:1 alignments, (c) is a 2:1 and (d) is a 3:1 alignment. On the left images we project matched phrases into the y axis to compute cratio$_y$. On the images on the right-hand side we project matched phrases into the x axis to compute cratio$_x$. The 2:1 alignment in (c) has the highest coverage score of these examples, since it has cratio$_x$ = 1.0 and cratio$_y$ = 1.0.

In general, the ratio of characters in $\boldsymbol{y}$ and $\boldsymbol{x}$ segments covered by the y-axis and x-axis projections of $\boldsymbol{T}$ is given by functions $\text{cratio}_y(\boldsymbol{T},\boldsymbol{y})$ and $\text{cratio}_x(\boldsymbol{T},\boldsymbol{x})$, respectively.

The implementation of these two functions is discussed ahead in Subsection 3.3.3.

For now, let us present the coverage score cscore equation:

$$\text{cscore}(\boldsymbol{T},\boldsymbol{x},\boldsymbol{y}) = \sqrt{\text{cratio}_x(\boldsymbol{T},\boldsymbol{x}) \cdot \text{cratio}_y(\boldsymbol{T},\boldsymbol{y})} \tag{3.1}$$

Function cscore receives three arguments: the set of phrase translations occurrences T generated in the previous step, and the two input segments, $\boldsymbol{x}$, and $\boldsymbol{y}$.

The output is a score between zero and one which is computed as the geometric average between the result of the two cratio functions.

3.3.3 Ratio of Characters Covered by Matched Phrase Translations

Functions $\text{cratio}_x(\boldsymbol{T},\boldsymbol{x})$ and $\text{cratio}_y(\boldsymbol{T},\boldsymbol{y})$ are similar. We will look at how to implement $\text{cratio}_x(\boldsymbol{T},\boldsymbol{x})$ and the implementation of $\text{cratio}_y(\boldsymbol{T},\boldsymbol{y})$ should be obvious afterwards. In the textual description that follows, we will give line numbers of corresponding pseudo-code in Figure 3.7.

```
   input  : list of rectangles T
   input  : segment x
   output : value between 0 and 1
 1 bounds ← empty list
 2 foreach rectangle t in T do
 3 |   (x_L, x_R) ← positions of left and right corners of t
 4 |   append (x_L, +1) to bounds
 5 |   append (x_R, −1) to bounds
 6 end
 7 sort bounds increasingly
 8 r ← 0
 9 c ← 0
10 foreach pair (x, i) in bounds do
11 |   if r = 0 then    (entering projected shadow)
12 |   |   x_L ← x
13 |   end
14 |   r ← r + i
15 |   if r = 0 then    (exiting projected shadow)
16 |   |   c ← c + x − x_L
17 |   end
18 end
19 return c/|x|
```

Figure 3.7: Algorithm for computing cratio_x

The input of $\text{cratio}_x(\boldsymbol{T},\boldsymbol{x})$ is a set of rectangles $\boldsymbol{T}$ and a segment $\boldsymbol{x}$.

Let *bounds* be an empty list (line 1). For each rectangle t in $\boldsymbol{T}$ we get the character-based positions (x_L, x_R) corresponding to the left and right corners of the rectangle, and we insert the pairs $(x_L, +1)$ and $(x_R, -1)$ into *bounds* (lines 3 to 5). The +1 indicates that a rectangle starts at x_L and the −1 indicates that a rectangle ends at x_R.

After iterating over all rectangles as described above, we sort *bounds* in increasing order (line 7).

Let r and c be two counters initialized to zero (lines 8 and 9).

For each pair (x, i) in *bounds*, we add i to r (line 14). If r was zero before adding i then it means that at position x we are entering the shadow projected into the x axis by rectangles in $\boldsymbol{T}$, and we save x as x_L (line 12). If r is zero after adding i then it means that at this position we are exiting the shadow projected into the x axis by rectangles in $\boldsymbol{T}$, and we add the distance between the previously-saved x_L and the current position to the number of covered characters c (line 16).

As we finish iterating over *bounds*, variable c will contain the total number of covered characters in segment $\boldsymbol{x}$. The ratio of characters covered by phrase translations is thus computed as:

$$\frac{c}{|\boldsymbol{x}|}$$

Where, $|\cdot|$ is an operator that returns the lengh of segment $\boldsymbol{x}$ in terms of characters.

As said earlier, the implementation of $\text{cratio}_y(\boldsymbol{T}, \boldsymbol{y})$ is analogous to the procedure above, but adapted to work on the y axis.

3.3.4 Alternate Alignment Configurations

As discussed earlier in the introduction of this chapter, the aligner must consider several alternate alignment configurations besides the 1:1 alignment hypotheses.

Now we will use Figure 3.6 as basis for explaining how the coverage-score varies for various alignment configurations involving the same sentences.

In this figure, we have four alternative alignments, (a), (b), (c) and (d). The alignment depicted in (c) is a 2:1 alignment between segments $[x_8, x_9]$ and $[y_8]$ and it has highest coverage score among the four alternative alignments. Note how the projections into the y and x axes, on the left- and right-hand sides of the Figure respectively, fully cover the segments.

By contrast, in the 3:1 alignment depicted in (d), between the segment $[x_8, x_9, x_{10}]$ and $[y_8]$, the cratio_x is 0.83 which brings cscore down to 0.91, even though this alignment contains all phrase translations contained in the 2:1 alignment depicted in (c) plus one, in (x_{10}, y_8). We can say that while (d) contains one more phrase translation than (c), it has in fact lower overall density of phrase translations, because most of x_{10} is not covered.

The 1:1 alignments depicted in (a) and (b) exemplify how phrase translations are distributed in a scenario where two sentences (x_8 and x_9) were merged into a single sentence (y_9) in translation, or the opposite. Looking at the y projections on the left-hand side of (a) and (b), we see that the covered areas are complementary, i.e. they do not

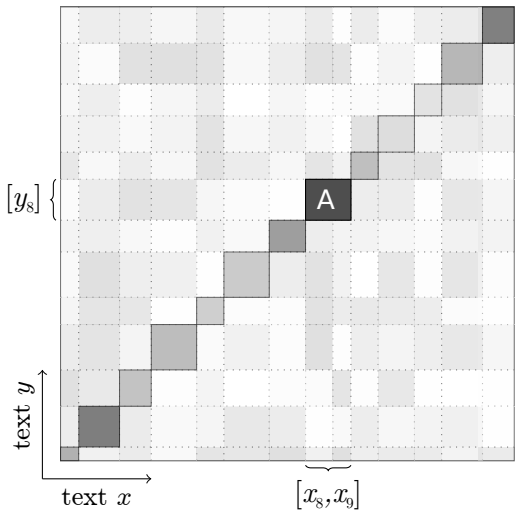


Figure 3.8: Coverage scores distribution in the sentence alignment hypothesis space. Recall that each small rectangle represents a 1:1 alignment hypothesis. Rectangles with darker shades have greater coverage score. True alignment hypotheses are outlined with solid lines (as opposed to dotted lines). Note the darker shades of the true alignments. Also, note the 2:1 alignment represented by rectangle **A**.

overlap. Also, looking at the *x* projections on the right-hand side of (a) and (b), we see that both projections are complementary.

When the projections of two contiguous alignment configurations are complementary, not necessarily as much as in this example, an aggregate configuration that subsumes both alignments, is likely to have greater coverage score, as is the case here, since (c) subsumes (a) and (b).

3.3.5 Turning Bleualign into a Coverage-Based Sentence Aligner

The Bleualign method adopts a commonly used dynamic programming to find a non-overlapping monotonic chain of parallel segments that *maximizes the total sum of scores* for all chained segments. This dynamic programming framework was first employed in the early length-based methods [15, 37] and has been adopted by most sentence alignment methods proposed afterwards.

In the original Bleualign implementation, aligned segments were scored with BLEU [88]. In my modified version, I replaced BLEU with the coverage-based score defined above. Apart from replacing BLEU score function with the coverage-based score function, I did not make any other substantial modifications to the Bleualign implementation.

Figure 3.8 presents the same alignment hypothesis space for a pair of short documents that was presented earlier in Figure 3.3, but here every 1:1 alignment hypothesis is filled with a shade of gray proportional to its coverage score; darker shades indicate greater coverage score.

Recall that the chain of rectangles running from the bottom left (the start of both

documents) to the upper right (the end of both documents) represents correctly aligned sentence pairs.

The coverage score is much higher (darker) for parallel sentence pairs than for non-parallel sentence pairs. This relatively high contrast between the darker shades of aligned parallel sentences and the lighter shades of the neighbouring non-parallel sentences, suggests that the coverage score has high discriminative power.

3.4 Evaluating Sentence Alignment

To evaluate sentence alignment performance we will compute precision, recall and F-measure for automatically generated alignments with respect to a gold-standard alignment.

As usual, precision is defined as the ratio of correct alignments over the number of generated alignments, recall is the ratio of correct alignments over the number of alignments in the gold standard, and F-measure is the harmonic mean of precision and recall.

Some alignments may be partially correct, as for example when the aligner proposes a 2:2 alignment where the reference contains instead two 1:1 alignments. Thus, following the practice from the original Bleualign evaluation [99], precision, recall and F-measure are reported according to a *strict* and *lax* criteria. In *strict* evaluation mode, only aligned segments that match exactly the reference are considered correct. In *lax* evaluation mode, aligned segments will be considered correct if they intersect reference segments on both language sides.

The coverage-based aligner was evaluated under exactly the same conditions as Bleualign was originally evaluated [99]: using the same gold-standard alignments and evaluation scripts. Furthermore, both aligners use the same phrase translation table which was obtained from the Europarl corpus [53] with Moses toolkit [55]. This table is part of the Moses 3.0 release and is available from the Moses website[6].

The coverage-based aligner uses the table directly while Bleualign uses it indirectly through Moses translation.

The Text+Berg corpus[7] is a German-French parallel corpus that has been manually aligned and is distributed with the source code of Bleualign[8]. The corpus is composed of yearbooks from Swiss Alpine Club and contains reports on mountain expeditions as well as some scientific articles. Because the domain of this corpus is quite different from the domain of the Europarl corpus, from which the phrase translation table was obtained, the performance of Bleualign and the coverage-based aligner are both likely to be lower than what could be achieved if both corpora were from the same domain. But despite

[6] http://www.statmt.org/moses/RELEASE-3.0/

[7] The Text+Berg corpus distributed with Bleualign seems to be only a part of the full Text+Berg corpus, but nevertheless, we refer to it as Text+Berg in this document.

[8] https://github.com/rsennrich/bleualign

Aligner	Strict			Lax		
	Prec	Rec	F-measure	Prec	Rec	F-measure
VANILLA	0.67	0.68	0.68	0.79	0.80	0.80
MBA	**0.86**	0.71	0.78	0.96	0.80	0.87
BLEUALIGN	0.83	0.78	0.81	0.98	0.92	0.95
COVERAGE	**0.86**	**0.84**	**0.85**	**0.99**	**0.96**	**0.98**

Table 3.1: Precision, recall and combined F-measure for alignments produced by four different aligners on the same gold standard corpus. Strict scores are obtained with strict comparison of alignments against the gold-standard. Lax scores count near misses as correct alignments.

this unfavourable scenario, both Bleualign and the coverage-based aligner perform better than the other aligners.

Table 3.1 shows the precision, recall and F_1 scores for the coverage-based aligner, Bleualign, and two well known aligners, the Microsoft Bilingual Aligner (MBA) by Moore [81] and the length-based aligner (vanilla) by Gale and Church [37].

In the strict evaluation mode, the coverage-based aligner outperforms all other aligners, especially in terms of recall.

For all aligners, the lax F-measures are about 10% higher than the strict scores, indicating a similar number of near-misses for all aligners, despite their different performances in terms of strict scores.

This could be explained by either one of two causes: (1) some regions of the gold standard may have been incorrectly aligned by human annotators, or (2) some regions of the gold standard are particularly hard to align. Unfortunately, to investigate if either case is true, one needs to known enough of German or French, which is not my case.

3.5 Summary

This chapter introduced a coverage-based score that, when used as a replacement for the BLEU score in Bleualign, improves the quality of alignments produced, particularly in terms of recall. Furthermore, compared to the original Bleualign, the modified version using the coverage-based score avoids the need to translate the texts prior to aligning them, which is time-consuming and error prone.

Although further experimentation with other language pairs is needed to draw stronger conclusions, experimental results obtained with a relatively dificult-to-align language pair, suggest that the coverage-based approach to sentence alignment may provide better performance than state-of-the-art aligners, provided that we are able to obtain a phrase translation table for the pair of languages under consideration.

These results have been peer-reviewed and published in the procedings of LREC 2016 [42].

The coverage-based sentence alignment method proposed here is an alternative to the lexicon-based alignment method proposed in my MSc thesis [39], allowing us to work with language pairs for which we do not have a large enough bilingual lexicon.

In future work, it would be interesting to investigate variations of the coverage-based score proposed here. More importantly, the evaluation should be extended to other language pairs and text domains. It would be interesting to participate, or perhaps co-organize a sentence alignment shared task in the vein of the WMT16 bilingual document alignment shared task [17].

Chapter Four

Phrase Alignment

4.1 Introduction

In the context of the phrase alignment problem, the term *phrase* denotes a sequence of words that may be translated as a unit, even if this sequence is decomposable into smaller individually translatable phrases. A single word is a phrase too.

Phrases may be split and written apart. For example, the phrase "we have decided" may be written apart by inserting one or more adverbs between the words "have" and "decided", as shown in Figure 4.1. In this example, "we have decided" is translated as a single Portuguese word, "decidimos", and thus it is not possible to align "we have" or "decided" separately.

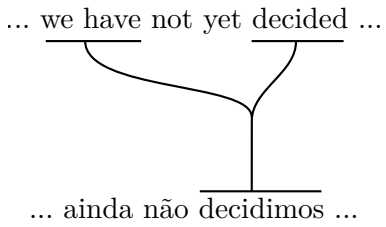


Figure 4.1: Alignment of a discontiguous phrase.

A phrase may occur multiple times in one sentence or document. Phrase alignment consists of identifying corresponding phrase occurrences within pairs of parallel sentences or documents. Thus, the input of a phrase alignment program is a pair of sentences, or documents, and the output is a set of *alignment links* between specific occurrences of phrases within the input sentences, or documents. As said earlier, in the thesis introduction, the final set of alignment links may be simply called the *alignment*.

Two phrases can only be correctly aligned (linked) if they are equivalent, but this condition, although necessary, is not sufficient. By pairing occurrences of phrases known to be translation equivalents we generate *alignment candidates*.

For example, consider the English and Portuguese sentences in Figure 4.2. The word "assessing" could, in principle, be aligned with either "considerar" or "a avaliação de",

since both these phrases are translation equivalents of "assessing". Similarly, the Portuguese word "considerar" could, in principle, be aligned with either "consider" or "assessing", since both these phrases are translation equivalents of "considerar". These three alignment candidates are represented as dotted lines in Figure 4.2.

If we choose the candidate that aligns "assessing" with "considerar", then we have no candidates left for "consider" or "a avaliação de". On the other hand, if we choose the candidate that aligns "consider" with "considerar", then we may still choose the candidate that aligns "assessing" with "a avaliação de". It turns out that these last two candidates are correct, while the first one is not.

There are two important observations to be made here: first, candidate choices are not independent of each other, and second, choosing correct candidates tends to leave less words unaligned than choosing incorrect ones.

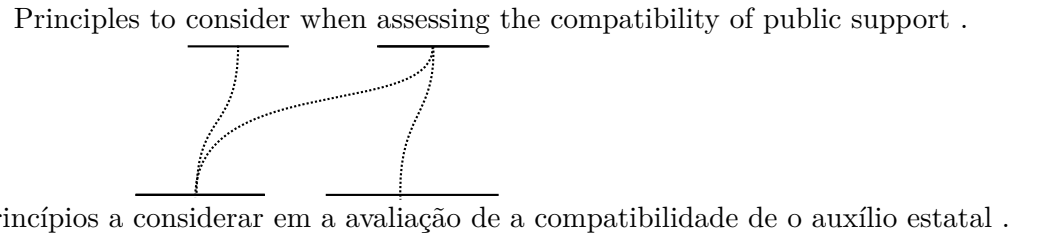


Figure 4.2: Multiple alignment alternatives for "assessing" and "considerar".

The number of possible alignments is much higher for function words, as exemplified in Figure 4.3. Since the preposition "to" may be translated as the Portuguese preposition "a", it can therefore be aligned with any of the three occurrences of the Portuguese word "a". Actually, only the first occurrence of "a" in the Portuguese sentence is a preposition, the other two occurrences are definite articles, but we assume that the machine does not know the part-of-speech of each word. Thus, we consider the three occurrences of "a" as candidate alignments for the preposition "to" as well as for the article "the".

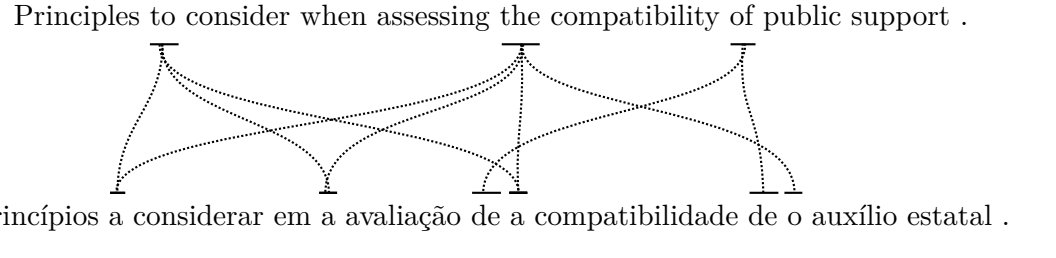


Figure 4.3: Multiple alignment alternatives for function words.

Depending on the language pair under consideration and the closeness of translations, there is a varying percentage of phrases that may be aligned monotonically, i.e. without crossings. Figure 4.4 shows how the pair of sentences that we have been using in the last examples is monotonically alignable almost on a word by word basis. When working with translations where a large percentage of phrases are monotonically alignable, like

in this example, a *monotonicity constraint* may be used as a filter to discard incorrect alignment candidates as most of the candidates shown in Figure 4.3. The use of such constraint was proposed in my MSc thesis [39], and before that, by António Ribeiro in his PhD thesis [94].

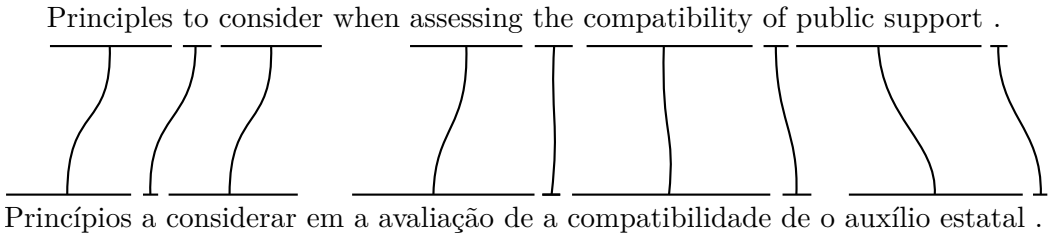


Figure 4.4: Example monotonic alignments obtained with the method proposed in my MSc thesis [39].

A less monotonic translation is shown in Figure 4.5. While these sentences are monotonically alignable for the most part, there is a long-range reordering which gives rise to "unaligned" segments in both sentences, as a result of the monotonicity constraint. In this particular example, the Portuguese sentence could have been written with exactly the same word ordering as the English sentence if the phrase "Apresenta-se de seguida" had been written after "planos" and adjusted to "é apresentada de seguida". In some language pairs, such as German-English, long-range reorderings similar to this one are very frequent.

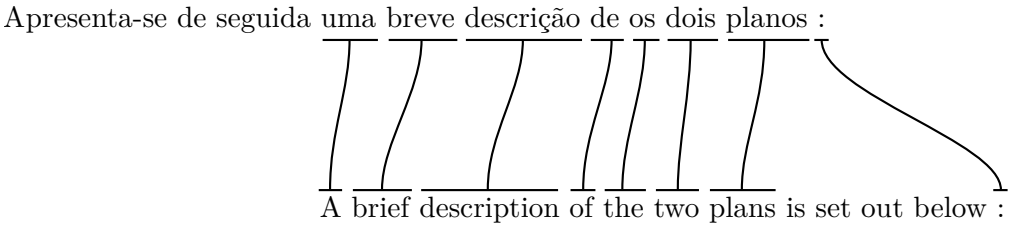


Figure 4.5: Example monotonic alignments obtained with the method proposed in my MSc thesis [39] for a pair of sentences with a long-range reordering. Segments that cannot be aligned monotonically were left "unaligned".

Another example of word reordering is given in Figure 4.6. Unlike in the previous example, here the sentences could not have been written in a closer word ordering, because English and Portuguese have different rules for composing noun phrases such as "European Public Assessment Report" and "Relatório Público Europeu de Avaliação".

There are other situations where phrases are even more strongly reordered in translation, either because the languages demand different word orderings or for stylistic preference of the translator. Thus, if an alignment method assumes and enforces monotonicity at all times, it will perform well for some texts but will struggle to align others.

Despite the crudeness of the monotonicity assumption in some situations, monotonic alignments produced by the method proposed in my MSc thesis [39] have been used by

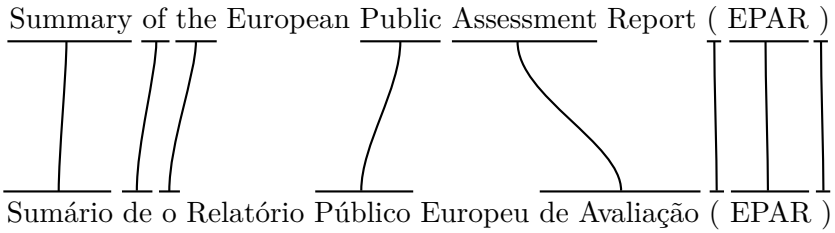


Figure 4.6: Example monotonic alignments obtained with the method proposed in my MSc thesis [39] for a pair of sentences with close-range reorderings.

José Aires, in the context of his PhD thesis [4], as the basis for building several Phrase-Based Statistical Machine Translation systems which perform significantly better than Moses systems [55] trained on the same corpora and lexica.

One pertinent question is if these state-of-the-art translation systems could be further improved by improving alignments in situations of non-monotonicity. The main goal of this chapter is to go beyond my previously proposed phrase alignment method [39] by allowing non-monotonic alignments, while ensuring that the precision of alignments does not decrease as a consequence of the increased generality. Achieving this goal will enable future work on improving the phrase-based machine translation system proposed by José Aires [4].

4.2 State of the Art

Several methods have been proposed for obtaining a phrase alignment of an input sentence-aligned parallel corpus. These methods generally follow one of three distinct approaches:

1. infer phrase alignments from word alignments;
2. joint learning of phrase alignments and translations;
3. phrase alignments based on previously extracted bilingual lexica.

These approaches are discussed in the next subsections, but we are primarily interested in the third one because it falls within the general goal of the thesis, which is to take advantage of previously extracted bilingual lexica.

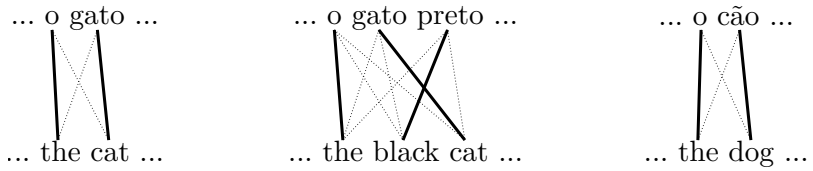
4.2.1 Phrase Alignments from Unsupervised Word Alignments

One of the earliest approaches, and still the most commonly followed, is to first align words using word-based statistical translation models, specifically the IBM models [16], and then infer phrase alignments based on those word alignments [32, 56, 108, 111, 116].

IBM Word-Based Translation Models

The first statistical machine translation models, developed at IBM by Brown et al [16] in the late eighties, were word based. While these word-based models have been superseded

(a) Initial state (all alignments are equally likely).



(b) Final state (some alignments are more likely than others).

Figure 4.7: Estimating word alignments with the EM algorithm. Initially all alignments are equally likely, but as the algorithm iterates, some alignments become more probable than others.

by phrase-based models for translation purposes, they are still widely used for obtaining initial word-level alignments, which in turn are used to infer phrase-level alignments.

The IBM models were not designed to take advantage of a bilingual lexicon. Instead, in these models, word alignments are learned jointly with word translations in an unsupervised manner from a sentence-aligned parallel corpus. Because word alignments and translations are both unknown, the learning algorithm considers hypothetical alignments between all words co-occurring in parallel sentences, which makes these methods computationally expensive.

Alignments are modeled as a hidden variable, which means that alignments are not directly observable in the training data, but nevertheless they are assumed to have an effect on the observable training data. Thus, in machine learning terms, we say that we have a problem of learning from *incomplete data*. The expectation maximization (EM) algorithm [30] is frequently employed in these scenarios, and Brown et al [16] devised the necessary mathematical treatment to apply EM to their models. Please refer to [16] for full details, or to [54] for a more pleasant description of the models and their training process. As the name suggests, the EM algorithm alternates over two steps:

- the *expectation* step applies the model to compute expectations for hidden data (alignments) taking into account the observable training data (parallel sentences) and the current model parameters (word translation probabilities);
- the *maximization* step takes the observable training data augmented with the predicted hidden data and updates the model parameters by computing maximum likelihood estimates (MLE).

The IBM models are a series of 5 increasingly complex models, which are *trained*[1]

[1] The *training* of a model in the context of *machine learning* is a process where the parameters of a model

sequentially with the EM algorithm, using the parameters of the last trained model to initialize the parameters of the next one to be trained. Model 1, being the simplest model, is the first to be trained and its parameters are initialized with a uniform probability distribution. Then, by iterating over the two steps, expectation and maximization, the model probabilities will converge (i.e. their variation decreases at each iteration). Figure 4.7 (a) illustrates alignments at the initial stage, where all alignments in every sentence are assumed to be equally likely (all lines representing alignments have the same strength). Figure 4.7 (b) illustrates alignments after several iterations of the EM algorithm, where some alignments have become much higher probability (stronger lines) than others.

At a fundamental level, this algorithm employs the model to predict the probability of each possible alignment and then adjusts the model parameters based on alignment counts weighted by these probabilities. As foreseeable, this feedback loop leads to amplification of some alignment errors, particularly for least frequent words as noted by Wang et al [114].

Extracting Phrase Alignments from Word Alignments

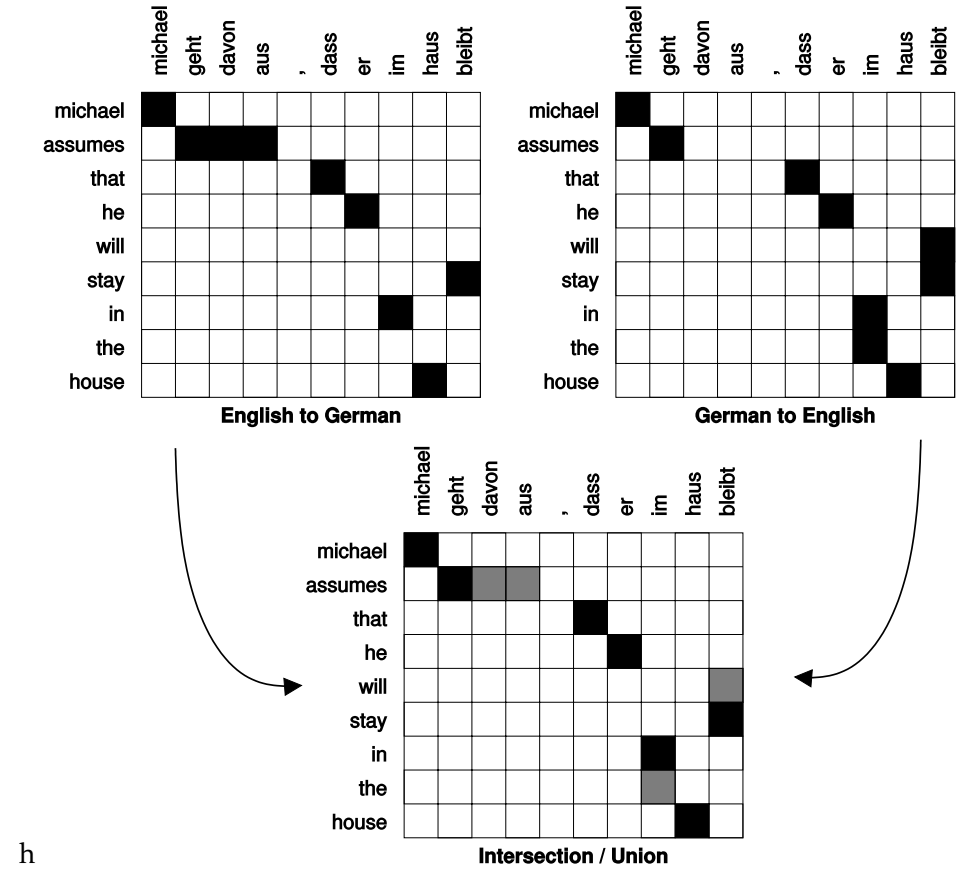
Several authors have proposed slightly different heuristics to extract phrase tables from word alignments [32, 56, 86, 108, 111]. Fundamentally, all these heuristics extract phrase pairs that are consistent with a set of word alignments. Since the IBM models are directional, it is established practice to compute alignments in both translation directions and intersect them to improve accuracy.

To reduce the number of unaligned words in the resulting set, some alignments are recovered from the union of both sets, by application of *growing heuristics*, for example by adding alignments that are neighbours to already accepted ones. This procedure is called *alignment symmetrization* and is depicted in Figure 4.8. Alignments belonging to the intersection of both directional sets are represented as black rectangles in the bottom half of the figure. Gray rectangles represent alignments recovered from the union of both directional sets. Note how the alignments for the German words “davon” and “aus” were recovered from the English to German alignments, and the alignments for the English words “will” and “the” were recovered from the German to English alignments. As an alternative to this post-alignment symmetrization procedure, Liang et al [64] proposed joint training of word alignments in both directions, which produces a more accurate symmetric alignment.

After alignment symmetrization, the phrase table extraction procedure will extract all phrase pairs, up to a specified maximum phrase lenght, that are consistent with the symmetrized alignment.

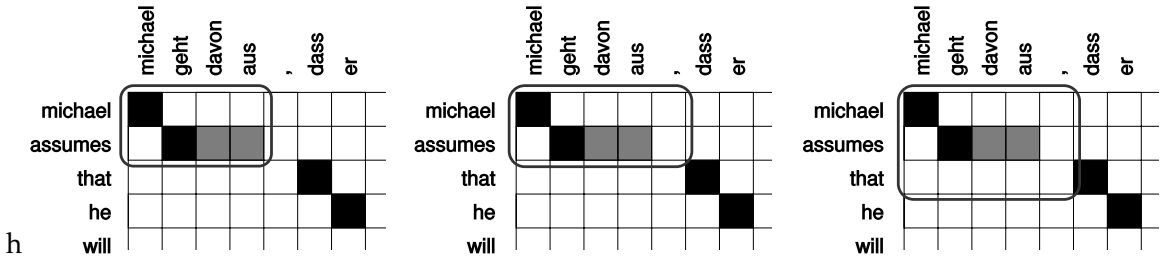
A pair of phrases are said to be *consistent* with the word alignment if every word in the source phrase is either aligned with words in the target phrase, or is unaligned, and

are adjusted to improve the accuracy of predictions for the given *training data*.



h

Figure 4.8: Word alignment symmetrization (adapted from [54]).



h

Figure 4.9: Phrase table extraction from word alignments (adapted from [54]). Phrase alignments are represented as rectangles with rounded corners. The alignments depicted on the left and the center are consistent with the word alignments but the alignment depicted on the right is not.

likewise, every word in the target phrase is either aligned with words in the source phrase or is unaligned.

Three example phrase alignments are depicted in Figure 4.9, represented by rectangles with rounded corners. At the left we have a simple case, where all words in the German phrase are aligned with a word inside the English phrase. At the center we have an alignment that includes an unaligned token (the comma in the German phrase), but this does not break consistency. At the right we have a an alignment that violates the

consistency rule because the word "that" in the English phrase is aligned with the word "dass" which lies outside the German phrase.

4.2.2 Joint Learning of Phrase Alignments and Translations

The previous approach is based on word translation models. Consequently, word alignment errors in multi-word expressions that cannot be aligned on a word by word basis are often propagated to the phrase-level alignments. A more direct alternative is to learn phrase alignments jointly with phrase translations [31, 32, 77, 83]. While the word-based IBM models described above consider hypothetical alignments between all pairs of words that co-occur in parallel sentences of the corpus, phrase-based unsupervised methods consider all contiguous phrases up to a maximum lenght in a corpus as possible alignment units, and all pairs of such phrases that co-occur in aligned sentences as alignment candidates. Because the number of distinct phrases is several orders of magnitude larger than the number of distinct words, training these models requires much more time and memory than the previous word-based approach, which is itself computationally expensive.

The method proposed by Zhang et al [117, 118] simultaneously segments sentences down to phrases and aligns them based on a statistical association measure, namely Point-wise Mutual Information (MI) instead of word alignments. Similar to our approach, presented ahead, this algorithm is a maximization algorithm, but in this case it maximizes joint probabilities, while in our approach we maximize the lexicon coverage. Furthermore, Zhang's algorithm is subject to a very strict consistency constraint: no word may belong to more than one phrase pair. This algorithm assumes that there is only one possible (minimal) phrase segmentation, while in our lexicon-based approach, multiple granularities are admissible in a hierarchical configuration. For example, we may align "president" with "presidente", "republic" with "república" and "republic president" with "presidente da república". All these three alignments are correct and useful. The Portuguese contraction of preposition and article "da" can only be aligned as part of the larger phrase. But the other two Portuguese words, "presidente" and "república", may be aligned with the corresponding English ones individually.

To mitigate the computational cost of training phrase-based models, many simplifications are made such as independence assumptions in the models, filtering of less frequent phrases, sampling and other cost-cutting measures. Thus, it is perhaps unsurprising, that the translation quality obtained by MT systems based on phrase alignments produced these methods is not significantly better than the quality of translations obtained with the previous approach. Moreover, Turchi et al [109] suggest that "it is unlikely that increasing dataset sizes will result in significant improvements" in the performance of unsupervised phrase-based models. Besides the arguments presented in their paper, which are anchored in a performance evaluation curve obtained by increasing training data size, this view is also supported by the lack of significant improvements since the phrase-based

models were first proposed in the late nineties and early two-thousands, with exception of the improvements brought by the paradigm shift into hierarchical models [20].

Like the word-based models described above, these phrase-based models were not designed to take advantage of a bilingual lexicon. Furthermore, only contiguous phrases are considered.

4.2.3 Lexicon-based Phrase Alignment

A third approach is to completely detach alignment from inference of translational equivalence of words or phrases. In this approach it is assumed that a bilingual lexicon of equivalent words or phrases is given as input to the alignment method, along with the parallel corpus [36, 39, 44, 105]. The main advantage of this approach is that we may exert quality control over the bilingual lexica used for alignment, which impacts the quality of alignments.

Contrasting with the first two approaches, where the aligner must infer an alignment for every target word/phrase, here in this 3rd approach, the aligner may leave some words or phrases unaligned, in particular, words and phrases not present in the input lexicon. We call this a *partial alignment*. Later, unaligned words and phrases will be targeted by word and phrase translation extraction methods, such as the ones presented in the next two chapters.

Lexicon- and Syntax-Based Phrase Alignment

Tambouratzis et al [105] propose a phrase alignment approach that is based on a bilingual lexicon, part-of-speech taggers and lemmatizers for both languages, and a target-language shallow parser and clause boundary detector. The parser is used to determine the phrasal segmentation of the target language which is then projected into the source language via matching of words, according to the input bilingual lexicon, and part-of-speech tags. The projection follows the restriction that phrases may not overlap in either language, which is similar to the consistency constraint of the algorithm proposed by Zhang [117, 118], discussed above. Tambouratzis' alignment algorithm has two steps: first, word-level correspondences are made on the basis of the input bilingual lexicon and, second, those correspondences are grouped into phrases depending on agreement of morpho-syntactic features provided by the taggers/lemmatizers.

Hierarchical Phrase Alignments Based on Word Lexicon

Flanagan [36] proposes a hierarchical phrase alignment method based on a bilingual lexicon of word translations. His method is hierarchical because it imposes a tree structure to the resulting alignments. The method has a *seed alignment generation* step followed by a *phrase alignment* step. Seed alignments are generated by looking up each word from the source sentence in the bilingual lexica and retrieving a list of possible translations,

which may be single words or phrases. Then, for each retrieved translation that happens to appear in the target sentence, it will generate a seed alignment between the source word and the translation. Note that Flanagan decided to restrict the matching on the source side to single words, although on the target side it will match phrases as well. This decision is not justified in the paper.

Besides the translations matched from lexica, the method will also generate seed alignments between identical tokens appearing in both sentences, such as punctuation, numbers and proper nouns. The *phrase alignment* step attempts at constructing a set of *consistent alignments* between phrases of both sentences that maximize the total score of seed alignments compatible with the aligned phrases. The method generates all possible source-target phrase pairs with reasonably proportional lengths within the parallel sentences and computes the score for each pair. Then, the phrase pairs with highest score are selected and all phrase pairs that happen to overlap with the selected ones are rejected. All the remainder pairs are rescored and the selection/rejection step is repeated. The algorithm ends when all pairs have been either selected or rejected.

4.2.4 Monotonic Coverage-Based Phrase Alignment

In my MSc thesis [39, 40], I proposed a coverage-based alignment method based on the hypothesis that among all monotonic subsets of a set of phrase alignment candidates, the most accurate should be the one with higher total coverage.

This method has a candidate *generation* step and a candidate *selection* step. The two steps are executed repeatedly in alternation, which results in an progressive refinement of the alignment.

At each iteration, the generation step is moderated and guided by the current alignment approximation, which is subsequently refined by the selection step.

Moderation in the candidate generation step is necessary because the method works at a document scope and thus generating candidates for all possible co-occurrences of known TEs would be prohibitively costly.

The candidate generation was based on exact matches of translation equivalents (TEs) from the input lexicon, which warrants high precision but sometimes leaves words phrases unaligned, even when these words or phrases are very similar to known TEs.

The final set of alignment candidates $\boldsymbol{C}$ was obtained by selecting a subset of alignment candidates with maximum total coverage, subject to a monotonicity constraint that forbids overlapping and crossed alignments.

The objective function to be maximized is:

$$\text{coverage}(\boldsymbol{C}) = \sum_{c_i \in \boldsymbol{C}} \text{len}_x(c_i) + \text{len}_y(c_i)$$

Where $\boldsymbol{C}$ is a set of candidates and $\text{len}_x(c_i)$ and $\text{len}_y(c_i)$ are the lengths of the alignment candidate c_i in the x and y dimensions, respectively.

The monotonicity or non-monotonicity of two candidates is determined by Equation 4.1.

$$\text{mono}(c_i, c_k) = \begin{cases} 1 & \text{if} \quad \text{end}_x(c_i) \leq \text{start}_x(c_k) \wedge \text{end}_y(c_i) \leq \text{start}_y(c_k) \\ 1 & \text{if} \quad \text{end}_x(c_k) \leq \text{start}_x(c_i) \wedge \text{end}_y(c_k) \leq \text{start}_y(c_i) \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Where $\text{mono}(c_i, c_k)$ is a boolean function that yields 1 (true) if candidate c_i precedes candidate c_k in both x and y documents or vice versa, or 0 (false) otherwise.

There are main limitations in this alignment method:

1. the monotonicity constraint employed in the selection step is sometimes unrealistic and rejects a large number of good candidates;
2. the candidate generation step is too cautious, and sometimes does not generate all candidates that should be generated, given the TEs in the lexicon;
3. alignment takes longer because the method works at document scope
4. the method does not support hierarchical alignments, non-monotonic alignments, nor discontiguous phrases.

Still, even with the limitations listed above, the alignments produced by this method were used by José Aires, in the context of his PhD thesis [4], as the basis for creating a series of phrase-based statistical machine translation systems that obtain, on average, 5.1 BLEU points above Moses systems [55] trained on the same corpora, for 8 language pairs[2] and 16 translation directions. Therefore, the monotonic coverage-based phrase alignment method should be considered state-of-the-art, despite its limitations.

4.3 Proposed Coverage-Based Phrase Alignment Method

In this section I propose a new phrase alignment method that is an evolution of the method proposed in my MSc thesis [39], described above and hereafter referred to as *the old method*. The main goal of the newly proposed method is to overcome the limitations in terms of fine-graininess and accuracy of alignments imposed by the monotonicity constraint of the old method.

4.3.1 Candidate Representation

Before we enter the description of the method itself, we will describe how candidates are represented, since the output of the method is a set of candidates.

A candidate is an object with at least three attributes: *id*, *tokens*, *coverage*. The *id* is an integer number that identifies each candidate within a set of candidates of a pair of sentences. The *tokens* attribute is a mapping of language identifiers, such as `"EN"` or `"PT"`, to lists of token positions within the sentences of the respective language. Token

[2] EN-DE, EN-FR, EN-ES, EN-PT, DE-PT, ES-PT, FR-PT and DE-ES

positions are zero-based, which means that the first token in a sentence occupies position 0. The *coverage* attribute is a real number that indicates the number of characters within the candidate that are known to be equivalent. The coverage is computed differently depending on how the candidate is generated and will be discussed in detail when we discuss candidate generation methods.

Figure 4.10 shows an example English-Portuguese pair of sentences, at the top, with some alignment candidates drawn between the sentences. Candidate number 11 aligns a discontiguous English phrase consisting of tokens 9 and 12 to Portuguese token 1. Candidates 15, 16 and 17 are arranged hierarchically, with candidate 16 subsuming candidates 15 and 17. Candidates 10 and 12 are non-monotonic with respect to candidate 11. None of these situations could be represented in the old monotonic alignment format. At the bottom we have the on-file representation of the alignment candidates using JSON format [3]. In JSON, objects are written as a list of comma-separated attributes enclosed within curly braces `{}`. An attribute is composed by an name, enclosed in quotes, followed by a colon and the respective value, which may be of any valid JSON type. A list is enclosed within square brackets `[]` and may contain zero or more values of any valid JSON type, separated by commas.

In the bottom half of Figure 4.10, each line starting with a curly brace followed by `"id":` defines a candidate. Only the three main attributes of candidates are shown (*id*, *tokens* and *coverage*). Compare the JSON representation of candidates 10 to 17 on the bottom of the figure with their graphical representation on the top half of the figure. The *id* of each candidate is draw at the left of the respective curved lines connecting English tokens to Portuguese tokens. Note how larger candidates tend to have higher coverage score.

4.3.2 General Idea

Like in the old method, there are two main steps: a candidate generation step and a candidate selection step. However, while in the old method we executed generation and selection steps repeatedly in alternation, which resulted in an iterative refinement of the alignment, here we execute generation followed by selection, only once.

The old method was iterative because it worked at a document scope, which required a progressive generation of candidates. The new method works at sentence scope, which is much smaller and thus allows generation of all possible candidates at once.

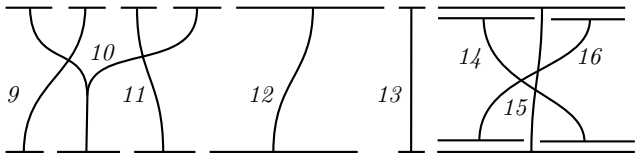
The candidate selection step is responsible for selecting a *consistent subset* of all candidates. A set of candidates is said to be consistent if each candidate in the set is consistent with all other candidates. Two candidates are consistent with each other if they do not overlap or if one subsumes the other, i.e. one stricly contains the other. A *hierarchical alignment* is a consistent set of candidates where each candidate is either subsumed by

[3] The JSON format is an easy-to-parse lightweight data format that has seen widespread usage in recent years. See `http://json.org/` for details.

Sentences, token positions and alignment candidates (not all)

```
EN: Possible interactions with other haematopoietic growth factors and cytokines
    0        1            2    3     4              5      6       7   8

    have not yet been investigated in clinical trials .
    9    10  11  12   13           14 15       16     17
```




```
PT: Não foram ainda investigadas , em ensaios clínicos , as possíveis interacções
    0   1     2     3            4 5  6       7        8 9  10        11

    com outros factores de crescimento hematopoiéticos e  com citoquinas .
    12  13     14       15 16          17              18 19  20         21
```

On-file representation of alignment candidates

```
"cands": [
    {"id": 0,  "tokens": {"EN": [0],      "PT": [10]   }, "coverage":  8.49, ...  },
    {"id": 1,  "tokens": {"EN": [1],      "PT": [11]   }, "coverage": 11.49, ...  },
    {"id": 2,  "tokens": {"EN": [2],      "PT": [12]   }, "coverage":  3.46, ...  },
    {"id": 3,  "tokens": {"EN": [3],      "PT": [13]   }, "coverage":  5.48, ...  },
    {"id": 4,  "tokens": {"EN": [4],      "PT": [17]   }, "coverage": 14.49, ...  },
    {"id": 5,  "tokens": {"EN": [5],      "PT": [16]   }, "coverage":  8.12, ...  },
    {"id": 6,  "tokens": {"EN": [6],      "PT": [14]   }, "coverage":  7.48, ...  }
    {"id": 7,  "tokens": {"EN": [7],      "PT": [18]   }, "coverage":  1.73, ...  },
    {"id": 8,  "tokens": {"EN": [8],      "PT": [20]   }, "coverage":  9.49, ...  },
    {"id": 9,  "tokens": {"EN": [10],     "PT": [0]    }, "coverage":  3.00, ...  },
    {"id": 10, "tokens": {"EN": [9, 12],  "PT": [1]    }, "coverage":  6.71, ...  },
    {"id": 11, "tokens": {"EN": [11],     "PT": [2]    }, "coverage":  3.87, ...  },
    {"id": 12, "tokens": {"EN": [13],     "PT": [3]    }, "coverage": 12.00, ...  },
    {"id": 13, "tokens": {"EN": [14],     "PT": [5]    }, "coverage":  2.00, ...  },
    {"id": 14, "tokens": {"EN": [15],     "PT": [7]    }, "coverage":  8.00, ...  },
    {"id": 15, "tokens": {"EN": [15, 16], "PT": [6, 7] }, "coverage": 15.49, ...  },
    {"id": 16, "tokens": {"EN": [16],     "PT": [6]    }, "coverage":  6.48, ...  },
    {"id": 17, "tokens": {"EN": [17],     "PT": [21]   }, "coverage":  1.00, ...  },
]
```

Figure 4.10: Representation of alignment candidates.

another candidate or subsumes one or more candidates. For example, candidates 14, 15 and 16, depicted in Figure 4.10, form a hierarchical alignment.

Like the selection step of the old method, the new selection step is based on the coverage-maximization hypothesis that we stated earlier in the introduction, and we repeat here: "the correct set of alignment links for a given problem instance should be the one, among all hypothetical sets of alignment links, that maximizes the coverage with respect to a bilingual lexicon." Remember that an *alignment link* is an alignment candidate that has been selected.

The main difference between the old and new selection steps is the type of constraint employed. In the old method we employed a monotonicity constraint, mainly because we were working at document scope and from such a broad perspective, phrases mostly follow a monotonic ordering given that phrases belong to sentences, which themselves follow a monotonic ordering, for the most part. Only when we look more closely, taking a single pair of sentences as alignment scope, the crudeness of the monotonicity constraint becomes more evident.

The new selection function is recursive and employs a simple, yet effective, non-overlapping constraint.

At sentence scope, the selection function will select a subset of non-overlapping candidates with maximum total coverage. Then, for each selected candidate c that subsumes at least two candidates we call the selection function recursively for the set of candidates subsumed by c.

The non-overlapping constraint ensures that the same character in a sentence is not counted as covered more than once at a given scope.

As a result of recursion we obtain hierarchical alignments such as candidates 14, 15 and 16 depicted in Figure 4.10.

Later in this section we will see in detail how the selection function is implemented as an integer linear programming (ILP) problem instance.

First, we will address candidate generation.

4.3.3 Candidate Generation

Candidate generation is accomplished by several *candidate generators*, which are functions that take the pair of sentences as input and return a set of candidates as output. There are two types of candidate generators: *ground generators* and *recursive generators*. Ground generators are used to generate an initial set of candidates that we call *first generation candidates* or *ground candidates*. Recursive generators are used to generate new candidates based on previously generated candidates. By applying recursive generators to the set of first generation candidates we would generate a *second generation* of candidates. Then, by applying recursive generators to the union of the first and second generations we obtain the *third generation*, and so on and so forth.

We will start by describing ground generators and then we move on to the recursive generators and how the two types of generators work together.

4.3.4 Matching Similar Tokens

The three simplest ground generators are responsible for generating alignment candidates for punctuation, numbers and cognates. They are called *match_punct*, *match_num* and *match_sim*, respectively.

The *match_punct* generator is responsible for generating candidates aligning occurrences of equivalent punctuation. For example, different languages employ different quotation marks. While single and double curly quotes, '…' and "…", are common in English, angular quotes «…» have been traditionally used in Portuguese, although the use of curly quotes in Portuguese is increasing. Other examples of equivalent but not identical punctuation are the Spanish inverted question and exclamation marks, "¿…?" and "¡…!", respectively, which should be aligned together with the normal question and exclamation marks. For example, when aligning an English question with a Spanish question, the inverted "¿" and normal "?" question marks at the begining and end of the Spanish question, respectively, would form a discontiguous phrase that would be aligned with the question mark at the end of the English question.

The *match_num* generator is responsible for generating alignment candidates for equivalent number formats. When different characters are used as decimal separators and thousands separators in the languages being considered, numbers will be written differently. For example, while in English we would write "10,000.00", in Portuguese we would write the same number as "10.000,00".

Finally, the *match_sim* generator is responsible for generating candidates for identical words such as proper nouns, and words with similar spelling, which are typically cognates. Measuring spelling similarity is addressed in Chapter 5, where we will define a function called *spsim*. For now, we only need to know that *spsim* will yield value 1.0 when two words are identical or when the differences between these words are safe to be ignored, meaning that the words should be considered as having equivalent spellings. Thus, *match_sim* will pair each word of one sentence with each word of the other and will generate a candidate for each pair that *spsim* yields 1.0.

4.3.5 Exact Lexicon Matching

The most important ground candidate generator is named *lookup* and it generates candidates for all co-occurrences of known TEs from the input lexicon.

This candidate generator requires one-time pre-processing of the input lexicon to create Aho-Corasick (AC) automata for phrases of both languages. We call this step *lexicon indexing* and it is depicted in Figure 4.11. This step is conceptually similar to the phrase translation indexing step employed in document and sentence alignment methods

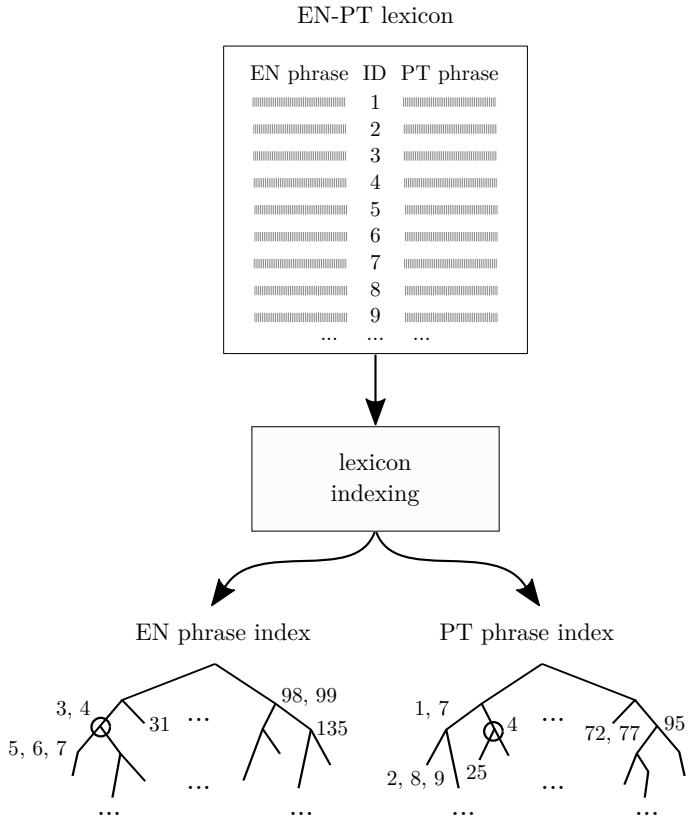


Figure 4.11: Lexicon indexing

proposed in the previous chapters, but here we are indexing a lexicon instead of a phrase translation table, and these two have very different characteristics.

A phrase translation table contains phrases of limited length, typically not longer than 7 tokens in the case of Moses [55]. By contrast, although most phrases in a bilingual lexicon are short, there is no enforced length limit. For example, we may have in the lexicon the Portuguese phrase "no sentido inverso ao dos ponteiros do relógio" which is a translation of "counterclockwise". This phrase, in the form that is usually written has 8 tokens. However, for enabling a better segmentation of phrases in alignment, we choose to expand all preposition-article contractions, which results in a phrase with 12 tokens: "em o sentido inverso a o de os ponteiros de o relógio".

Another important difference between a phrase translation table and a bilingual lexicon is that, for any phrase in the table, it is likely that all or most sub-phrases of that phrase are also in the table. A bilingual lexicon is much more sparse. For example, we may have "o Presidente de a República" in the lexicon but we will not have "o Presidente de a".

As shown in the figure, the lexicon indexing procedure creates two AC automata, one for phrases of each language. An AC automaton is a tree-shaped data structure where phrases are encoded as a path descending from the root node, represented at the top of the trees of Figure 4.11. A more complete description of the AC automaton is given in

Appendix A.

The tree node associated with each known phrase contains a list of TE identifiers associated with that phrase. For example, the nodes enclosed by circles in the trees of Figure 4.11 encode the two phrases that make up the TE with ID 4. From this representation we see that the phrase encoded by the node in the English tree is associated with TEs 3 and 4, while the phrase encoded by the node in Portuguese tree is associated only with TE 4.

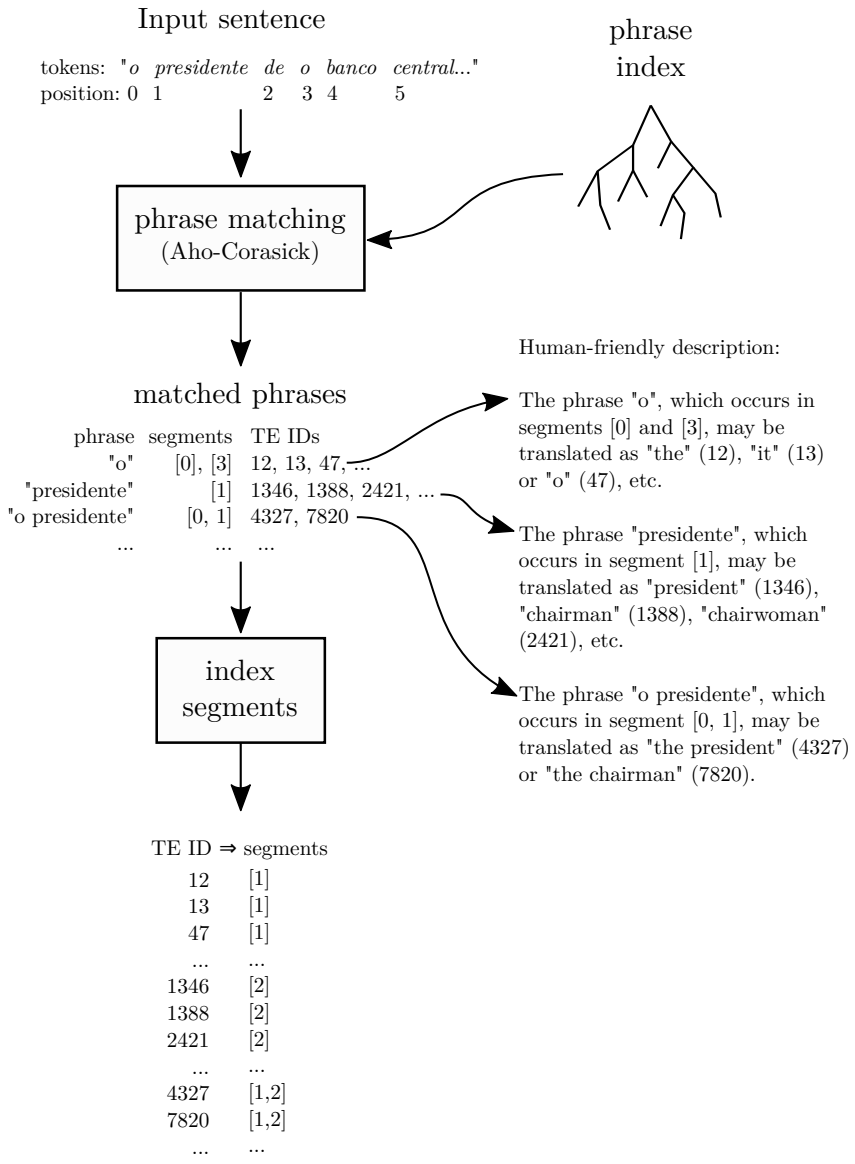


Figure 4.12: Matching known phrases and creating segment indices.

The lookup procedure begins by applying AC automaton to efficiently locate all occurrences of known phrases of arbitrary lenght in each sentence, as depicted at the top of Figure 4.12. As a result we obtain a list of matched phrases, where for each distinct phrase matched we have a list of segments where it occurs, and a list of TE IDs associated

with that phrase. Remember that a segment is a list of token positions. Thus, segment `[1]` corresponds to the phrase (word) "presidente" in the example of the figure, while segment `[0,1]` corresponds to phrase "o presidente".

The next step is to create a hash table, mapping TEs to segments, which we call a *segment index*, as depicted in the lower part of the figure. To create a segment index we iterate over each matched phrase and for each TE ID we insert it into the index as key and the list of matched segments as the associated value.

Finally, the generation of candidates is performed by taking the segment indices created in the previous step, for both sentences and for each TE ID that is present in both indices we generate candidates for all combinations of associated segments.

As an example, supose that a given TE ID is associated with segments `[6], [14], [19]` in the English index and with segments `[7], [13]` in the other.

Then we would generate the following 6 candidates:

```
{"id": 1, "tokens": {"EN": [6],  "PT": [7]},  ... },
{"id": 2, "tokens": {"EN": [6],  "PT": [13]}, ... },
{"id": 3, "tokens": {"EN": [14], "PT": [7]},  ... },
{"id": 4, "tokens": {"EN": [14], "PT": [13]}, ... },
{"id": 5, "tokens": {"EN": [19], "PT": [7]},  ... },
{"id": 6, "tokens": {"EN": [19], "PT": [13]}, ... },
```

The coverage of an alignment candidate c for a pair of sentences $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by Equation 4.2.

$$\text{coverage}(c, \boldsymbol{x}, \boldsymbol{y}) = \sqrt{\left(\sum_{i \in \text{seg}_x(c)} |\boldsymbol{x_i}| \right) * \left(\sum_{j \in \text{seg}_y(c)} |\boldsymbol{y_j}| \right)} \tag{4.2}$$

Where seg_x and seg_y are functions that return the segments of the candidate in each language, $\boldsymbol{x_i}$ is the token at position i in sentence $\boldsymbol{x}$ and $\boldsymbol{y_j}$ is the token at position j in sentence $\boldsymbol{y}$. As usual, $|w|$ is the lenght, in terms of characters, of word w.

This coverage score is the geometric mean of the total number of characters covered by the candidate in each language.

4.3.6 Matching Discontiguous Phrases

A discontiguous phrase is a phrase that may appear in text as two or more separate subphrases. For example, the phrase "we have decided" may appear intermixed with adverbs, as in "we have not yet decided". At the places where a phrase may be interrupted by other phrases, we say that we may have a *gap*, which we represent by the special token $*, as in "we have $* decided". This special token indicates that at its position in a phrase we may find zero or more words that do not belong to the phrase.

In Chapter 7 we will see how a large number of discontiguous phrases may be inferred automatically from the bilingual lexicon. Here we will assume that our input bilingual

English	ID	Portuguese
we have $* analysed	1	analisámos
we have $* decided	2	decidimos
we have $* decided	3	tomámos uma decisão
we will $* analyse	4	analisaremos
we will $* decide	5	decidiremos
we will $* decide	6	tomaremos uma decisão

Table 4.1: Mini English-Portuguese lexicon containing discontiguous phrases

lexicon already contains discontiguous phrases and we will focus on how to efficiently match them in sentences.

To this end, I employ a data-structure based on the AC automaton, which I call a *treetree*. A treetree is a tree containing multiple AC automatons inside it. All automatons are isolated from each other by edges labeled $*. Figure 4.13 depicts a treetree that encodes the English phrases of the mini-lexicon in Table 4.1.

The IDs of TEs associated with each English phrase are enclosed in rectangles in the rightmost nodes.

In this example there are three AC automatons, enclosed within gray-shaded circles.

At this point, if the reader is not familiar with the AC automaton, it is recommended to read Appendix A before proceeding.

The matching procedure of a treetree is similar to the matching procedure using an AC automaton: we start at the root of the tree, and, as we consume tokens from the input sentence, we try to descend in the tree[4].

The only difference to the normal Aho-Corasick algorithm happens when we reach a node that has an outgoing edge labeled $*: at this point, we create a new *cursor* rooted at the tree pointed by the edge, and we carry on with the normal AC matching algorithm. The special $* token is never seen at the input.

A *cursor* is a tiny data structure that stores a pointer to a tree node and a list of positions of matched tokens.

As an example, let us search the sentence "we have not yet decided what to do" for matches of the phrases encoded in the treetree of Figure 4.13.

We will use Table 4.2 to help us visualize the state of the matching algorithm as we consume the sentence token by token.

We start at the root of the tree, which is node 0 and thus far we have not matched any tokens. Thus our initial cursor is the pair (0, []).

As we consume the token "we" at position 0 from the input sentence, we descend the cursor to node 1 through the edge labeled "we". The cursor is now (1, [0]) indicating that we are at node 1 in the tree and we have, thus far, matched the token at position 0 in the input sentence.

[4]In Figure 4.13 we represented the tree horizontally, instead of vertically, to save space. Thus, to *descend* in this tree we start at the leftmost node and we move to the right.

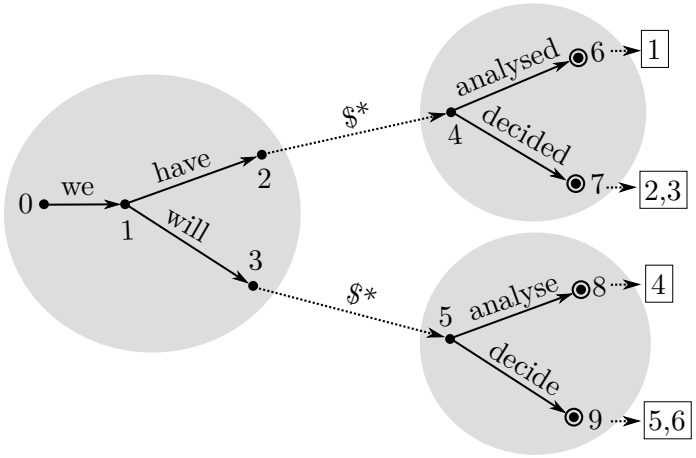


Figure 4.13: An example treetree

Current Cursors	Input Pos	Input Token	Output
(0,[])	0	"we"	
(1,[0])	1	"have"	
(2,[0, 1]), (4, [0, 1])	2	"not"	
(0,[]), (4, [0, 1])	3	"yet"	
(0,[]), (4, [0, 1])	4	"decided"	([2, 3], [0, 1, 4])
(0,[]), (7, [0, 1])	5	"what"	
(0,[]), (4, [0, 1])	6	"to"	
(0,[]), (4, [0, 1])	7	"do"	

Table 4.2: Step by step state changes when matching a short sentence with a treetree.

Next, we consume the token "have" at position 1 from the sentence and we descend to node 2. Since there is an outgoing edge labeled $*, we immediately create a new cursor, pointing at node 4, with the list of tokens matched thus far, which is [0, 1].

As we now have two cursors, the next input token will be consumed by each cursor separately.

When we consume the token "not" at position 2 with the first cursor, which is pointing at node 2, we cannot descend because there is no edge labeled "not". Thus we follow the failure link of that node (not drawn in the figure) which points to the root node. Given that we moved from node 2 which had depth 2 into the root node, which has depth 0, we remove the last 2 matched token positions from the cursor. Thus, the first cursor now becomes (0, []). The second cursor, which currently points at node 4 will not change as a result of consuming the token "not" because there is no outgoing edge from node 4 with label "not" and this node is a root node of a inner tree.

Next, we will consume the token "yet" at position 3 of the input sentence. In this case, both cursors will stay unchanged, because there are no outgoing edges labeled "yet" from the current cursor nodes.

Next, we consume the token "decided" at position 4 of the sentence. Again, the first cursor will not be updated because there are no outgoing edges from node 0 labeled

"decided". However, from the node of the second cursor we now descend to node 6, which is an *output node*, meaning that we have matched a discontiguous phrase. As a consequence of arriving at this node, we output the pair ([2, 3], [0, 1, 4]), where [2, 3] is the list of TE identifiers associated with the matched phrase, and [0, 1, 4] is the matched segment.

Next, we consume the token "what" at position 5 of the sentence. Once again, the first cursor will not be updated because there are no outgoing edges from node 0 with label "what". In the case of the second cursor, which now points at node 7, we will follow the failure link (not drawn in the figure) pointing to node 4, because there is no outgoing edge labeled "what" from node 7.

Next, as the remainder tokens are consumed, the cursors will rest at nodes 0 and 4 because none of these tokens is a label of an outgoing edge from those nodes.

Conceptually, having multiple cursors open is the same as having multiple AC automatons running in parallel on separate regions of the treetree. The number of open cursors grows as we advance in the input sentence, but sentences are usually short enough that the number of cursors is not a problem in terms of computational cost. On average, there are 7.5 cursors open at the end of a sentence. On longer sentences (100 tokens or more) the average increases to 42 cursors open at the end of each sentence.

The generation of candidates for TEs with discontiguous phrases is identical to the generation of candidates described in the previous subsection, except for the data structure used to index phrases.

If there are discontiguous phrases in the input lexicon, we use treetrees to index phrases, else we use plain AC trees. The creation of treetrees is identical to the creation of a regular AC tree up to the point when failure links and output links are created. Thus, the decision wether the tree being created at the lexicon indexing step will be a treetree or a regular AC tree is made at the instant when the program encounters the first discontiguous phrase in the input lexicon.

The creation of failure links and output links in a treetree follows exactly the same algorithm that would be used for a regular AC tree, within each subtree delimited by $* edges. In the example of Figure 4.13, this would mean that the creation of failure and output links would be made as if each tree enclosed within a gray-shaded circle was an independent AC tree.

4.3.7 Stem-Based Lexicon Matching

This subsection describes a candidate generator named *stem_lookup* which is based on *lookup*.

The purpose of *stem_lookup* is to generate alignment candidates from inexact matches of lexicon TEs based on word stems instead of full word forms.

Another purpose of *stem_lookup* is to align compound words which are not yet in the lexicon themselves, but are composed by stems for which we already have translations in

Example Input Lexicon		Stemmed Lexicon		Newly Matched Phrases	
English	**Portuguese**	**English**	**Portuguese**	**English**	**Portuguese**
abbreviated	abreviada	abbrevi	abrevi	*abbrevi*ated	*abrevi*adas
abbreviated	abreviados			abbreviated	abreviado
				abbreviated	abreviaram
				abbreviated	abreviou
acclaimed	aplaudiu	acclaim	aplaud	acclaimed	aplaudida
acclaimed	aplaudido			acclaimed	aplaudidas
acclaimed	aplaudidos			acclaimed	aplaudiram
accounted for	tida em conta	account for	tid em cont	accounted for	tidas em conta
				accounted for	tido em conta
				accounted for	tidos em conta
european	europeia	european	europ	european	europeias
european	europeu				

Table 4.3: Example English-Portuguese lexicon, the corresponding stemmed version and phrases that can be matched with the stemmed version but not with the original lexicon.

the lexicon.

Like the *lookup* generator described in the previous subsection, the *stem_lookup* also needs to perform a lexicon indexing step before alignment proper takes place. However, before the lexicon indexing, there is yet another step, which is to prepare a derived version of the bilingual lexicon from the original input lexicon.

This derived version is obtained by replacing every word in the lexicon by its stem. We call the resulting lexicon a *stemmed lexicon*.

Table 4.3 showns an example English-Portuguese lexicon on the two left-hand side columns and the resulting stemmed lexicon is shown on the two center columns. It stands out from this table that a stemmed lexicon typically has many fewer entries than the original lexicon.

To get the stem of each word we use off-the-shelf rule-based stemmers, specifically those from the Snowball project [89, 90, 98]. Since each language has its own morphology rules, stemmers are forcibly language dependent, thus limiting the applicability of this option to languages for which stemming rules have been written.

To get a sense of the usefulness of *stem_lookup* consider the phrase pairs shown in the two right-hand side columns of Table 4.3, which do not exist in the example input lexicon, but which *stem_lookup* is able to match using the stemmed lexicon shown in the two center columns.

After creating the stemmed lexicon we execute the lexicon indexing step that was described above in the *lookup* generator and which will create two AC trees, which we will call the *stemmed lexicon indices*.

These trees are token-based, which means that they are designed to efficiently find known phrases within sentences, but they cannot helps us locate stems within compound words. For that purpose we need to create two character-based AC trees, one for each

language, and in these trees we will index only individual stems, not stemmed phrases.

These two character-based AC trees are called *stem indices*.

Let us now move on from the pre-processing step onto the procedure for generating candidates for a given pair of sentences.

The first step is to create stemmed versions of the sentences by replacing each word with its stem. These stemmed sentences are then processed with the *lookup* method, described above, but using the stemmed lexicon indices instead of the normal lexicon indices.

At the end of the *lookup* invocation, we keep the candidates generated and the segment indices created within lookup. Remember that these indices are hash tables mapping TE IDs into matched segments in each sentence. Segment indices will be needed to generate candidates for compound words.

The next step is to apply the character-based AC trees to locate all occurrences of known stems within each word in each sentence. A stem occurrence is represented by a tuple of three integers: (1) the word position in the sentence, (2) the character-based position where the occurrence starts within the word, and (3) the character-based position where the occurrence ends within the word. For each stem occurrence found we check if any of the TE IDs associated with that stem is present in the segment index of the other language. For each TE ID found in the index, we retrieve the associated list of segments and we generate candidates by pairing the stem occurrence with each segment.

Candidates generated from stem occurrences will have an extra attribute named `chars` containing the character-based positions of the beginning and end of the stem within the word.

To make this explanation easier to follow, let us consider the following English-Portuguese pair of sentences:

```
workplace security          segurança no local de trabalho
0         1                 0         1  2     3  4
```

In addition, let us assume that by applying the English character-based AC tree we found the following two stem occurrences in the English sentence: (0, 0, 3) and (0, 4, 8). The occurrence (0, 0, 3) indicates that a known stem was found within token 0 starting at character 0 and ending at character 3, which corresponds to substring "work". Next, lets suppose that by consulting the Portuguese segment index, we find segment `[4]` associated with the same TE ID that is associated with occurrence (0, 0, 3). Then, we would generate the following candidate:

```
{
    "id": 1,
    "tokens": {
        "EN": [0],
        "PT": [4]
    },
    "chars": {
        "EN": [[0,3]]
    },
    "coverage": 5.66
}
```

Note that while previously we have written each candidate object in a single line of JSON, here we display a candidate object with identation, for easier understanding. Also, because the *tokens* and *chars* attributes are themselves objects with attributes, we will use the usual dotted path notation to navigate this object attribute hierarchy. Thus, *cand.tokens* refers to the attribute *tokens* of the object *cand*, and *cand.tokens.EN*, refers to the attribute *EN* of the object *cand.tokens*.

To compute the coverage of the candidate above, we need the length of the matched English stem, which is $3 - 0 + 1 = 4$. We also need the length of the matched Portuguese word at position 4, which is 8. Therefore the coverage of this candidate is $\sqrt{4 * 8} = 5.66$.

The occurrence (0, 4, 8) indicates that a known stem was found within token 0 starting at character 4 and ending at character 8, which corresponds to substring "place". Next, lets suppose that by consulting the Portuguese segment index, we find segment `[2]` associated with the same TE ID that is associated with occurrence (0, 4, 8). Then, we would generate the following candidate:

```
{
    "id": 2,
    "tokens": {
        "EN": [0],
        "PT": [2]
    },
    "chars": {
        "EN": [[4, 8]]
    },
    "coverage": 5
}
```

The length of the matched stem is $8 - 4 + 1 = 5$ and the length of the Portuguese word at position 2 is 5. Therefore the coverage of this candidate is $\sqrt{5 * 5} = 5$.

Note that the attribute *cand.chars.EN* is a list of pairs of character-based positions, i.e it is a list of two-element lists. The attribute *cand.chars.EN* must have exactly as many elements as the list *cand.tokens.EN*, or be ommitted. Likewise, attribute *cand.chars.PT* must have exactly as many elements as the list *cand.tokens.PT*, or be ommitted, which is the case in the example candidates aboves. When *cand.chars.EN* is ommitted, all English tokens referred in *cand.tokens.EN* are assumed to be fully matched. Likewise for *cand.chars.PT* and *cand.tokens.EN*.

To summarize, the *stem_lookup* generator applies a pre-processing function that replaces words by their stems. This pre-processing is applied to TEs when indexing the lexicon and to input sentences when generating candidates.

Besides generating candidates by matching phrases of stems instead of words, *stem_lookup* also generates candidates based on stems that occur within compound words.

The coverage of candidates generated by *stem_lookup* is computed taking into consideration the length of matched stems instead of the full words. Thus, if both *lookup* and *stem_lookup* generate candidates with exactly the same segments, then the candidate generaged by *stem_lookup* will have lower coverage than the candidate generated by *lookup* because stems are shorter than full words.

Whenever two candidates are generated with exactly the same segments on both languages, the candidate with lower coverage is discarded.

4.3.8 Recursive Candidate Generation

The candidate generators presented thus far are *ground generators* because they generate candidates without taking into consideration other candidates.

A *recursive generator* is a candidate generator that generates new candidates based on previously generated candidates.

The algorithm for candidate generation, which combines ground and recursive generators, is presented in Figure 4.14.

In line 3 we iterate over ground generators discussed in the previous sub-sections. Specifically:

1. the *lookup* generator
2. the *stem_lookup* generator
3. the *match_punct* generator
4. the *match_num* generator
5. the *match_sim* generator

Each of these generators is invoqued in line 4 and generated candidates are added to C.

At this point, C contains the first generation of candidates.

From line 6 to 12 we have a loop that will be repeated until C becomes empty. At each iteration of this loop, a new generation of candidates will be generated by recursive generators. In line 7, the previous generation of candidates is added to C_{out} and in the next line, C is emptied. In line 9 we iterate over recursive generator functions and on

```
   input  : pair of sentences (s_x, s_y)
   output : set of candidates C_out
 1 C_out ← empty set
 2 C ← empty set
 3 foreach ground generator function G do
 4 |   C ← C ∪ G(s_x, s_y)
 5 end
 6 while C is not empty do
 7 |   C_out ← C_out ∪ C
 8 |   C ← empty set
 9 |   foreach recursive generator function R do
10 |   |   C ← C ∪ R(s_x, s_y, C_out)
11 |   end
12 end
13 return C_out
```

Figure 4.14: Candidate generation using ground and recursive generators.

line 10 each generator is called. Note that the invocation of a recursive generator takes an additional parameter compared to the invocation of a ground generator in line 4. The extra parameter is the set of all candidates generated in previous generations, which we have stored in C_{out}. When no new candidates are generated, C becomes empty and the procedure terminates.

There are two recursive generators:

1. *pat_match*, which generates candidates by matching translation patterns
2. *mono_cat*, which generates new candidates by concatenating monotonically adjacent candidates

The first generator relies on translation patterns, which will be addressed in Chapter 6. Therefore, the description of the method how translation patterns are matched is given in that chapter. The second recursive generator is described next.

4.3.9 Concatenating Monotonically Adjacent Candidates

Since there is a body of work ([4, 5, 6, 24, 26, 39, 94]) based on monotonic alignments attesting the usefulness of monotonicity as an approximation for several language pairs, it would be unwise to completely discard such a good approximation.

Thus, in this new alignment method we will not abandon monotonicity as a whole. Instead, we change the role of monotonicity from being an enforced constraint into a preference, which is expressed by means of generating larger candidates, with slightly higher coverage, by concatenating monotonically adjacent candidates as exemplified in Figure 4.15.

At the top of the figure we have first generation candidates that were generated by the ground generators discussed above. These candidates will be passed as input for the first

invocation of *mono_cat*, which generates new candidates by concatenating monotonically adjacent candidates.

Right below the first generation candidates we have represented the second generation of candidates, resulting from the invocation of *mono_cat*. For example, candidate c_{10} results from the concatenation of c_1 with c_2, c_{11} results from the concatenation of c_2 with c_3, and so on and so forth.

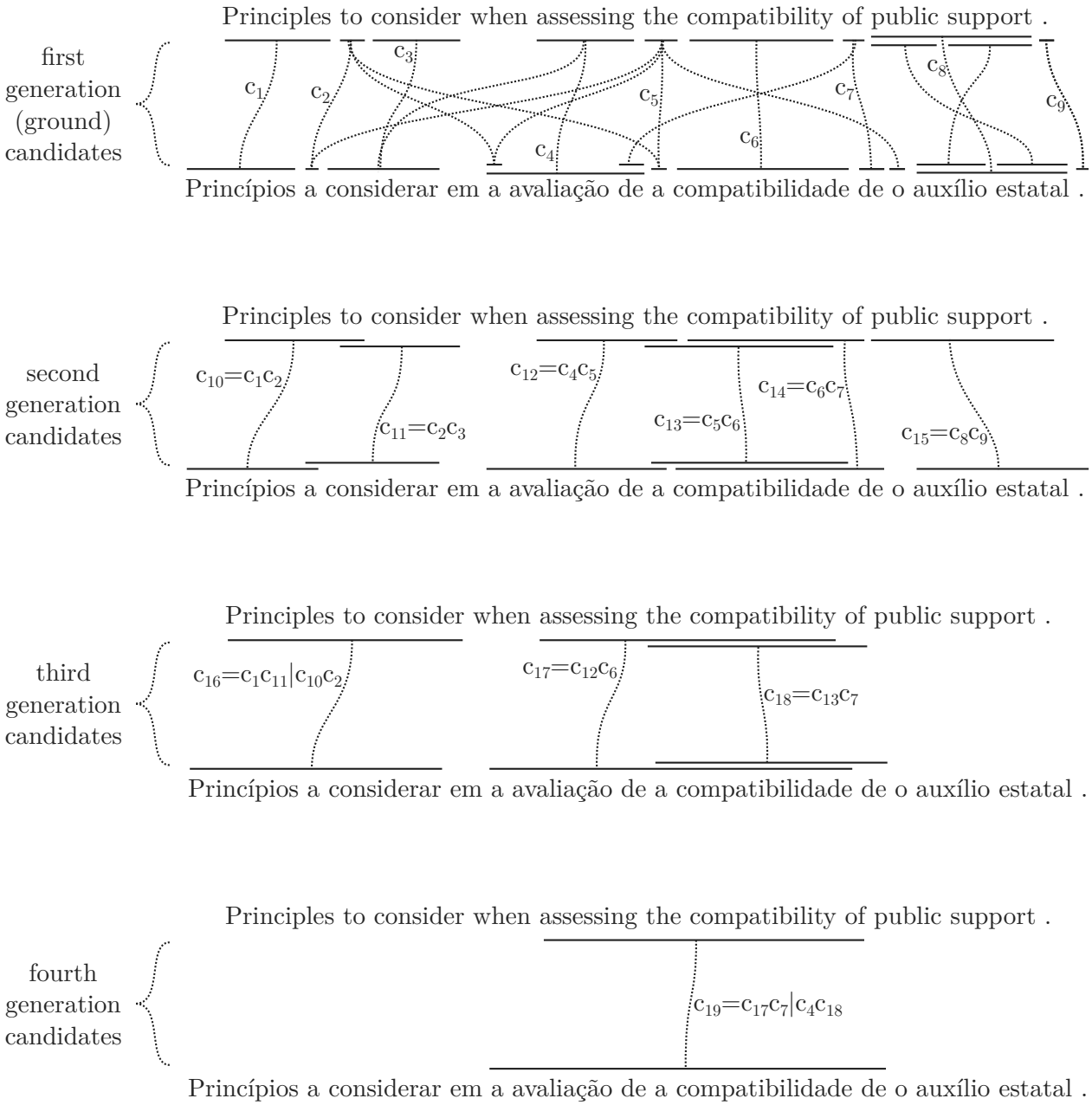


Figure 4.15: Example candidates resulting from concatenation of monotonically adjacent candidates. For simplicity, not all first generation candidates are numbered; only those that are concatenated in subsequent generations.

Candidates resulting from concatenation have slightly higher coverage than the sum

of coverages of concatenated candidates because the whitespace in-between the concatenated candidates is counted as being covered. Thus, the coverage of a candidate resulting from the concatenation of candidates c_i and c_k is $= \Big(\text{coverage}(c_i) + \text{coverage}(c_k) + 1\Big)$.

When a *mono_cat* candidate is selected by the selection step, described next, the candidates that were concatenated together to produce it are also immediately selected as part of the final set of alignment candidates. After the selection step, all candidates produced by *mono_cat* are discarded. The purpose of *mono_cat* candidates is to bias the selection step to prefer monotonic sequences of alignments when they exist, by creating candidates with slightly higher coverage.

For better understanding the purpose of these candidates let us consider the four candidates for the word "the" show at the top of Figure 4.15. Of these four candidates, only c_5 is correct, but all candidates have the same coverage score. Thus, if we applied the selection step at this point, considering only the first generation candidates, there would be at least four consistent subsets of candidates with the same total coverage score, resulting from selecting any of the candidates of word "the". Now let us see what happens when we generate concatenated candidates.

By concatenating candidates c_5 and c_6 we generate the second generation candidate c_{13} which has slightly greater coverage than the sum of coverages of c_5 and c_6. Thus, if we applied the selection step at this point, giving the union of first and second generation candidates as input, it would select c_{13}, which implicitly selects c_5 and c_6, instead of selecting c_6 with any of the other three candidates for "the".

4.3.10 Candidate Selection

This subsection describes how candidate selection is implemented as an integer linear program (ILP) that maximizes the total sum of coverage scores of the selected candidates subset. The following objective function translates coverage-maximization algebraically:

$$\text{maximize} \sum_{i}^{N} x_i \text{coverage}(c_i)$$

Where N is the total number of generated candidates, i is the identifier of a candidate, x_i is a binary variable indicating if the candidate c_i is selected or not, and $\text{coverage}(c_i)$ returns the coverage score of candidate c_i.

The maximization will be subject to a set of constraints, expressed as linear inequations. We want to avoid selecting candidates that overlap. Thus, for each unordered pair of candidates c_i and c_k from the set of generated candidates, if $\text{overlap}(c_i, c_k)$ returns true, then we add the following constraint to our ILP instance:

$$x_i + x_k \leq 1$$

Since x is a vector of variables that can only take value 0 or 1, the inequation above can only be true if either candidate i or candidate k is not selected. If both were selected, the inequation would not be true, and thus it would not be a solution to this ILP instance.

After selecting candidates at a sentence scope, we will apply the selection step recursively to smaller scopes defined by the segments of candidates that strictly subsume two or more candidates. This recursion will enable the generation of hierarchical alignments. If a candidate stricly subsumes only one candidate, then the subsumed candidate can be immediately accepted because there are no competitor candidates to reject it.

The current implementation is able to use the COIN-OR [5] CBC solver or the GLPK [6] solver, whichever happens to be installed in the computer. The performance of both of these solvers is comparable.

4.3.11 Conversion to Monotonic Alignment

Because there are several tools that were developed expecting the monotonic alignment format produced by the old aligner, we need to convert the new alignments into the old format, for compatability with those tools.

Obviously, the conversion process will be lossy because, as discussed earlier, the old format does not support non-monotonic alignments, hierarchical alignments, nor discontiguous phrases.

The conversion is straightforward: from the set of selected candidates, we will select a monotonic subset with maximum total coverage.

The monotonic selection is implemented using a ILP formulation similar to the one employed in the non-monotonic selection described above.

There are two differences: the monotonic selection is not recursive, and the constraints are based on monotonicity of candidates, instead of the overlapping condition.

For each unordered pair of candidates c_i and c_k from the set of selected candidates, we check if c_i and c_k are monotonically consistent with each other, according to the function $mono(c_i, c_k)$, defined earlier in Equation 4.1. If $mono(c_i, c_k)$ returns false (zero) then the two candidates are not monotonically consistent with each other we add the following constraint to our ILP instance:

$$x_i + x_k \leq 1$$

Remember that in our ILP problem formulation, x is a vector variable with dimension equal to the number of candidates and each element can only assume integer values. Thus, the constraint defined above will prevent candidates i and k to be selected simultaneously.

The monotonic alignment format resulting from the conversion of the alignment candidates represented earlier in Figure 4.10 is shown in Figure 4.16. In this format, only

[5] https://www.coin-or.org/

[6] https://www.gnu.org/software/glpk/glpk.html

Sentences and token positions (for reference)

```
EN: Possible interactions with other haematopoietic growth factors and cytokines
    0        1            2    3     4              5      6       7   8

    have not yet been investigated in clinical trials .
    9    10  11  12   13           14 15       16     17



PT: Não foram ainda investigadas , em ensaios clínicos , as possíveis interacções
    0   1     2     3            4 5  6       7        8 9  10        11

    com outros factores de crescimento hematopoiéticos e  com citoquinas .
    12  13     14       15 16          17              18 19  20         21
```

Monotonic alignment as shown by the concordancer

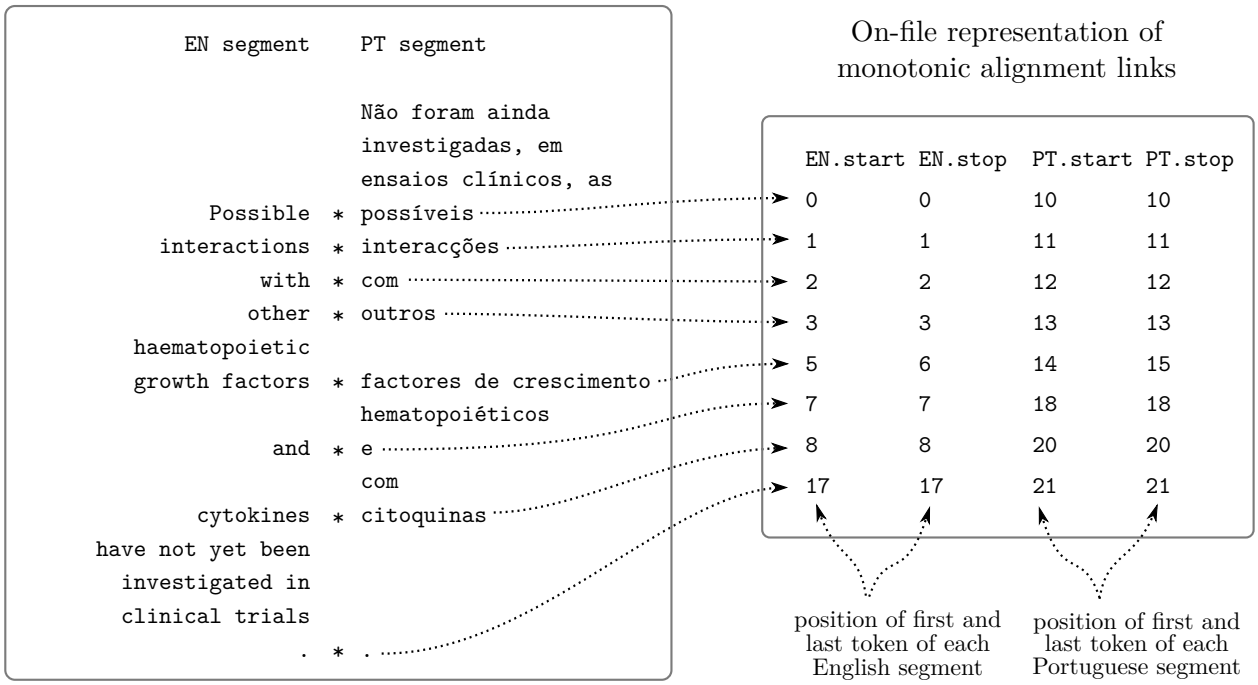


Figure 4.16: On-file representation of monotonic alignments, after conversion.

the positions of the first and last tokens of each segment are written to the output file, which is why discontiguous phrases are not fully supported[7].

In this example, only 8 of 17 candidates were preserved in the conversion process, which means that more than half of the information was lost in the conversion.

[7] If a candidate with a discontiguous segment is selected by the monotonic selection of the conversion procedure, it will be written into the output as if it was a contiguous segment, i.e. including all tokens between the first and the last tokens in the segment.

4.4 Evaluating Phrase Alignments Quality

To evaluate the quality of phrase alignments we measured the quality of translations produced by a phrase-based statistical machine translation system trained on those alignments. This is an indirect evaluation of alignment quality, based on the assumption that better alignments lead to better translations.

We used the translator developed by José Aires in the context of his PhD thesis[4]. This translator was designed to work with the monotonic alignments produced by the method proposed in my MSc thesis [39], which we consider as baseline. Therefore, we had to convert the new alignments into the older monotonic format, losing all non-monotonic, hierarchical and non-contiguous alignments in that process. Only a monotonic subset of alignment links was preserved.

For this reason, the translation quality obtained with the new alignments converted into the old monotonic format is not expected to be drastically different from the quality obtained with the baseline alignments.

4.4.1 Training and evaluation corpora

The evaluation was carried out over four language pairs, specifically DE-PT, EN-PT, ES-PT and FR-PT, in both translation directions for each language pair. We used three parallel sub-corpora of different sizes and domains which are part of the OPUS corpus [107]. These corpora are described below.

For evaluating the translation quality of the systems trained with each corpus, we created a corresponding testset, composed of sentence-aligned documents belonging to the same domain, but that are not included in the training corpus.

Initial sentence alignments of the testset documents were obtained with the method described in Chapter 3. Afterwards, these alignments were manually checked and fixed as needed.

The EU constitution corpus is the smallest of the three corpora used, occupying approximately one megabyte per language. This corpus consists of a single large document: the unratified international treaty which was commonly designated as European Constitution. As testset for the systems based on the EU constitution corpus, we sentence aligned translations of the consolidated version of the Treaty of Nice[8]. See Table 4.4 for details about the size of the EU constitution corpus and the respective testset.

The ECB corpus occupies approximately 40 megabytes per language and is composed of web pages and documents downloaded from the European Central Bank website[9]. The testset for the systems based on the ECB corpus was obtained by aligning translations of the 2015 ECB Annual Report, which is not included in the ECB corpus. The jargon employed in these highly technical documents associated with a higher than usual translation freedom, made manual verification of sentence alignments harder. For the language

[8] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:C:2016:202:TOC

[9] https://www.ecb.europa.eu/

Language	Training Corpus Size			Evaluation Corpus Size		
	Sentences	Tokens	Characters	Sentences	Tokens	Characters
DE	8962	137997	980043	871	15153	105472
PT (DE)	8962	126997	724647	871	19684	109927
EN	9989	158677	925110	871	18127	104443
PT (EN)	9989	136509	782000	871	19565	109524
ES	10185	173577	1025292	869	19239	113063
PT (ES)	10185	140910	807058	869	19534	109351
FR	10468	180594	1009953	864	20010	112085
PT (FR)	10468	141540	807053	864	19317	108119

Table 4.4: Sizes of the EU constitution corpus, used for training the PBSMT systems, and the Treaty of Nice translations, used for evaluation.

pairs DE-PT and ES-PT, I was not able to manually check and fix sentence alignments to the degree of accuracy expectable for a reference set of translations. Therefore, for the ECB corpus, only two language pairs were evaluated, specifically EN-PT and FR-PT. See Table 4.5 for details about the size of the ECB corpus and the respective testset.

Language	Training Corpus Size			Evaluation Corpus Size		
	Sentences	Tokens	Characters	Sentences	Tokens	Characters
EN	202011	5830072	32917138	4384	52249	315374
PT (EN)	202011	6853422	37883263	4384	65200	378992
FR	202875	7320855	40104581	1552	37435	214042
PT (FR)	202875	6928354	38295954	1552	34589	200749

Table 4.5: Sizes of the ECB corpus, used for training the PBSMT systems, and the 2015 ECB Annual Report translations, used for evaluation.

The EMA corpus[10] is the largest of the three corpora, occupying approximately 90 megabytes per language. It was compiled from PDFs downloaded from the European Medicine Agency (EMA) website [11]. As a testset for the systems based on the EMA corpus, we manually checked and fixed sentence alignments for translations of 10 European Public Assessment Reports (EPAR) into four language pairs, totalling 50 PDF documents arranged in 40 pairs. EPARs are small documents, with an average of 60 sentences each. We checked that these EPARs are not included in the EMA corpus. See Table 4.6 for details about this corpus size.

To summarize, we trained a total of 40 PBSMT systems for all combinations of 2 alignment methods, 4 language pairs, 2 directions and 3 corpora, except for the ECB corpus, where only 2 language pairs were considered.

[10]The European Medicine Agency (EMA) was previously known as the European Agency for the Evaluation of Medicinal Products (EMEA).

[11]http://www.ema.europa.eu/

Language	Training Corpus Size			Evaluation Corpus Size		
	Sentences	Tokens	Characters	Sentences	Tokens	Characters
DE	1101553	12813423	87998990	607	11443	77502
PT (DE)	1101553	16488583	94303829	607	13455	76865
EN	1082135	13864392	80350755	579	11086	63241
PT (EN)	1082135	16284623	93159873	579	13231	75376
ES	1097500	15833706	92821924	586	12647	74336
PT (ES)	1097500	16452168	94087480	586	13420	76574
FR	1105655	17396235	99958978	583	14478	82767
PT (FR)	1105655	16477642	94219141	583	13316	75889

Table 4.6: Sizes of the EMA corpus, used for training the PBSMT systems, and the translations of 10 EPARs, used for evaluation.

	Languages	Alignment Type		Delta
	Source→Target	Baseline	New	
h	PT→DE	28.33	29.80	+1.47
	DE→PT	38.82	39.79	+0.97
	PT→EN	53.72	58.35	+4.63
	EN→PT	51.58	55.81	+4.23
	PT→ES	59.39	60.46	+1.07
	ES→PT	60.02	60.09	+0.88
	PT→FR	59.59	61.81	+2.22
	FR→PT	60.73	61.83	+1.10

Table 4.7: BLEU scores (%) of translations produced by PBSMT systems trained on the EU constitution corpus and evaluated on the Treaty of Nice translations.

4.4.2 Evaluation Results

The BLEU scores of the translations produced by the systems trained on the EU constitution corpus are presented in Table 4.7

In this corpus we obtained consistent improvements over the baseline systems. The biggest improvement was observed for the EN-PT language pair, with 4 BLEU points. One possible explanation for this deviation, is the fact that this particular language pair and corpus were often used for quick experiments when developing extraction methods. As a result, the EN-PT part of the EU constitution corpus is likely to have much greater coverage from the lexicon than the other language pairs.

The BLEU scores of the translations produced by the systems trained on the ECB corpus are presented in Table 4.8. Again, we observe a consistent improvement across all language pairs in both translation directions, although the BLEU deltas are less expressive than in the EU constitution corpus and the absolute BLEU scores are much lower than in the EU constitution corpus. As said above, ECB annual reports contain a considerable

	Languages Source→Target	Alignment Method Baseline	New	Delta
p	PT→EN	34.57	34.95	+0.38
	EN→PT	32.32	32.89	+0.57
	PT→FR	29.03	30.20	+1.17
	FR→PT	33.30	34.06	+0.76

Table 4.8: BLEU scores (%) of translations produced by PBSMT systems trained on the ECB corpus and evaluated on the 2015 ECB Annual Report translations.

	Languages Source→Target	Alignment Method Baseline	New	Delta
h	PT→DE	20.88	20.47	-0.41
	DE→PT	28.21	28.39	+0.18
	PT→EN	41.16	42.21	+1.05
	EN→PT	39.62	39.57	-0.05
	PT→ES	41.82	41.49	-0.33
	ES→PT	41.31	42.18	+0.87
	PT→FR	35.43	36.95	+1.52
	FR→PT	37.51	38.52	+1.01

Table 4.9: BLEU scores (%) of translations produced by PBSMT systems trained on the EMA corpus and evaluated on translations of 10 EPARs.

amount of jargon and sometimes the translations are too loose. Also, this corpus was never used to extract TEs before this experiment. This combination of factors explains the lower BLEU scores obtained in this corpus.

The BLEU scores of the translations produced by the systems trained on the EMA corpus are presented in Table 4.9. In this corpus we got BLEU scores higher than those obtained on the ECB corpus, but lower than those obtained on the EU constitution corpus. Unlike in the other two corpora, here we did not obtain consistent improvements across all language pairs and translation directions. For three out of four language pairs there was an improvement in one translation direction and a worsening in the other. Overall, the improvements are generally greater than the worsenings. Only for the FR-PT pair we got improvements in both translation directions.

Given that the alignments are symmetric, i.e. they are the same irrespective to translation direction, we may only attribute the signal fluctuations in the relative performance of alignments produced by the two algorithms to the heuristics used by the translator in the training process, when it extracts the translation phrase table. Hopefully, these heuristics can be improved in future work, while adapting the translator to use the newer alignments.

To summarize our findings, in 37 out of 40 translation directions, we observed an

Languages	MT System	Biological		Health	
		BLEU %	Δ	BLEU %	Δ
EN → PT	ISTRION-BOX	17.55	+2.17	19.01	+1.79
	Moses baseline	15.38		17.22	
PT → EN	ISTRION-BOX	20.88	+3.29	21.50	+3.02
	Moses baseline	17.59		18.48	

Table 4.10: Official results of the WMT16 biomedical translation shared task for the English-Portuguese language pair in the biological and health sub-domains.

improvement in translation quality of 1.20 points, on average. In the remainder situations, there was a slight loss, of 0.26 BLEU points on average.

Thus, we conclude that the new alignments, even after a lossy conversion process, still retain greater precision than the baseline.

4.4.3 Participation in the WMT16 biomedical translation task

Besides the comparative evaluation presented above, the alignment method proposed has also been used to align the English-Portuguese corpus provided by WMT16 biomedical translation shared task organizers [13]. These alignments were then used as the basis for creating the ISTRION-BOX phrase-based statistical machine translation (PBSMT) system that participated in the WMT16 biomedical translation shared task [6], having achieved 1.8 to 3.3 BLEU points above the Moses baseline, trained over the same corpora, as shown in Table 4.10.

These BLEU scores were computed by the shared task organizers. The participants did not have access to the reference translations during the evaluation campaign. Thus, the results presented in Table 4.10 are an independent confirmation of the advantages of our lexica-based approach, even though most of our lexicon was extracted from european legislation, which is quite different from the biomedical domain.

Initially we expected to achieve higher BLEU scores overall, but after a brief analysis of the corpus we concluded that there were several corpus-quality issues that could explain the lower-than-expected scores:

1. The provided corpus seems to have a higher number of incorrectly aligned sentences than is usual in other corpora such as the ones used in the experiments above;
2. The corpus has some documents written in Brazilian Portuguese and others written in European Portuguese;
3. Of the documents written in European Portuguese, some follow the new spelling rules introduced in the Portuguese language orthographic agreement of 1990 while others follow the older spelling rules.

A quantification of these corpus-quality issues would fall outside the scope of this thesis, and was not carried out.

Even though most of our English-Portuguese lexicon had been extracted from european legislation written in European Portuguese and not from the biomedical domain being considered in this experiment, we still surpassed the Moses baseline by 2 to 3 BLEU points. This demonstrates that the knowledge acquired from one domain can be transferred to other domains effortlessly with our lexicon-based approach to alignment. Furthermore, we believe that the translation quality obtained with our system would further increase after a few iterations of the global workflow described in the thesis introduction, but the time-constrained nature of the shared task did not allow for this kind of experimentation.

4.5 Summary

In this chapter I proposed a new phrase alignment method which evolved from the method proposed in my MSc thesis [39]. Specifically, the new method has the following improvements:

1. non-monotonic alignments;
2. hierarchical alignments;
3. discontiguous phrases;
4. inexact matching of phrases and words.

Despite the increased generality of this new method, the alignments produced have been demonstrated to be at least as accurate as the alignments produced by the method earlier proposed in my MSc thesis [39].

Because the decoding engine used in the machine translation (MT) evaluation experiment has not yet been adapted to take advantage of the new richer alignments, we can justifiably expect improvements in the translation quality when those changes are made to the decoder, in future work.

The phrase alignments produced by this method have also been used, after convertion to the old monotonic format, for constructing bilingual concordances in the context of Jorge Costa's PhD thesis [23].

Furthermore, this phrase alignment method also served as the foundation for implementing the extraction method described in Chapter 6.

Chapter Five

Extraction Based on Spelling Similarity

In the previous chapter we addressed the problem of phrase alignment, making use of a bilingual lexicon containing word and multi-word translation equivalents. In this and the next chapter we will address the complementary problem of extracting new translation equivalents from sentence- and phrase-aligned texts.

We will start by exploiting the spelling similarity of words that share a common etymology, called *cognates*[1].

5.1 Introduction

Many language pairs share cognates, either because the languages have a common ancestry, or because words were imported from each other or from other languages. As an example, consider the cognates "father" (English), "vader" (Dutch and Afrikaans), "Vater" (German), "pater" (Latin), "patar" (transliterated Sanskrit), "padre" (Spanish and Italian), "patra" (transliterated Greek), "père" (French), "pai", (Portuguese and Galician), and "far" (Swedish and Norwegian). Like this, many other words have permeated multiple languages giving rise to wide etymological trees of cognates.

Given that cognates typically retain some degree of spelling similarity to the original word form, the spelling similarity has been widely used as an indicator of cognaticity and, by extension, of translational equivalence between words of different languages [12, 79, 94, 95, 101].

Not all words with similar spelling are cognates, though. For example, despite the high similarity between the English word *mill* and the Portuguese word *mil*, which means *thousand*, they originate from different Latin words, *molinum* and *mīlle*, respectively, and thus are not cognates. Also, not all cognates have equivalent meanings. For example,

[1] from Latin *cognatus*, from *co-*'together with' + *natus* 'born'

while English *mile* and Portuguese *mil* both originate from Latin *mīlle*, they have different meanings nonetheless. Independently of having the same origin or not, if two words belong to different languages and have similar spelling and/or sound but different meanings, we call them *false friends*.

Fortunately, because false friends have different meanings they also tend to occur in different contexts. This means that false friends do not co-occur as often in parallel sentences as equivalent cognates do. Therefore, a reliable method for cognate extraction should combine a spelling similarity measure with a co-occurrence similarity measure, such as the Dice coefficient, which has been shown by António Ribeiro to be one of the most accurate measures among 28 tested for the task of extracting translation equivalents [96].

While this chapter is focused on the problem of measuring spelling similarity, this does not imply that spelling similarity is more important, or a replacement, for co-occurrence similarity measures and other features, such as phonetic similarity, suggested by Kondrak and Sherif [58]. Instead, our goal in this chapter is to take advantage of human-validated translation equivalents (TEs) to create similarity measures specially adapted to each language pair.

Ideally, a similarity measure adapted for English-Portuguese, would be able to ignore the character-wise spelling differences between "normally" and "normalmente", since we could say that suffix "ly" is equivalent to suffix "mente", and thus these two words have equivalent spellings. In this chapter, I propose a language-independent method for automatically generating an adapted similarity measure from a list of previously validated TEs.

5.2 State of the Art

The most commonly used measures of spelling similarity for the task of cognate identification, such as the longest common sub-sequence ratio (LCSR) and edit-distance-based similarity (EdSim) measures [12, 79, 94, 95, 101], treat all spelling differences in the same way, disregarding the fact that cognates exhibit spelling differences that are recurrent in each pair of languages. For example, "ph" and "f" consistently appear in corresponding positions in English and Portuguese cognates such as "phase"↔"fase" or "telephone"↔"telefone". Such recurrent spelling differences are often due to the evolution of spelling rules within each language, and thus they should not be penalized (for the purpose of identifying cognates) in the same way as arbitrary differences. For example, while "pharmacy" is translated as "farmácia" according to today's Portuguese spelling rules, before 1911 it was written as "pharmácia", which was closer to the original Greek word "pharmakeia".

We say that a spelling similarity measure is *static* if it does not adapt itself to take into account recurrent spelling differences like "ph"↔"f" of each language pair being considered. Conversely, we say that a spelling similarity measure is *adaptable* if it is able

to incorporate knowledge about recurrent spelling differences of each language pair and change its behaviour accordingly.

5.2.1 Static measures

Many different measures have been proposed in literature to assess the spelling similarity between words. The most basic one is perhaps the 4-character-prefix rule introduced by Simard et al [101] that yields as hypothetical cognates any pair of words that share a common prefix of at least 4 characters.

LCSR

A more sophisticated similarity measure is the longest common sub-sequence ratio (LCSR), introduced by Melamed [79], which is given by the equation:

$$\mathrm{LCSR}(w_1, w_2) = \frac{|\mathrm{LCS}(w_1, w_2)|}{\max(|w_1|, |w_2|)} \tag{5.1}$$

where w_1 and w_2 are the words under consideration, LCS is the longest (not necessarily contiguous) character subsequence that is common to both words, and $|w|$ denotes the number of characters in the word w.

EdSim

A similar measure was suggested by Ildefonso and Lopes [48], based on the edit distance between words. Since the authors did not give a name to that measure, we will refer to it as EdSim, standing for edit-distance-based similarity measure, when we use it as a baseline measure in our evaluation experiment, later. EdSim is given by the equation:

$$\mathrm{EdSim}(w_1, w_2) = 1 - \frac{\mathrm{ED}(w_1, w_2)}{\max(|w_1|, |w_2|)} \tag{5.2}$$

where $\mathrm{ED}(w_1, w_2)$ is a function that returns the *edit distance* between w_1 and w_2. The edit distance is the minimum number of character-wise *edit operations* needed to transform one word into the other. There are three possible edit operations: *insertion* of one character, *deletion* of one character, and *substitution* of one character by another.

For example, to transform the English word "common" into the Portuguese equivalent "comum" we need to delete one "m" from the English word, replace the second "o" by a "u" and the "n" by an "m". Thus, the edit distance between "common" and "comum" is 3. Replacing the value of ED in Equation 5.2 by 3 and the lengths of "common" and "comum" by 6 and 5, respectively, we get $\mathrm{EdSim}(\text{"common"}, \text{"comum"}) = 1 - 3/\max(6, 5) = 0.5$.

The edit distance between two non-empty words w_1 and w_2 is defined recursively as:

$$\mathrm{ED}(w_1, w_2) = \min \begin{cases} \mathrm{ED}(\mathrm{pref}(w_1), w_2) + 1 \\ \mathrm{ED}(w_1, \mathrm{pref}(w_2)) + 1 \\ \mathrm{ED}(\mathrm{pref}(w_1), \mathrm{pref}(w_2)) + \mathrm{mismatch}(\mathrm{last}(w_1), \mathrm{last}(w_2)) \end{cases}$$

Where last(w) is a function that returns the last character of a word w, and pref(w) returns the prefix of word w up to but not including the last character. Function mismatch(c_1, c_2) is a character comparison function defined as:

$$\text{mismatch}(c_1, c_2) = \begin{cases} 0 & \text{if} \quad c_1 = c_2 \\ 1 & \text{otherwise} \end{cases}$$

Since ED is defined recursively, and one or both words are shortened by one character at each recursive invocation, we need to handle the recursion base cases, when one or both words are empty, denoted as nil:

$$\begin{aligned} \text{ED}(w_1, nil) &= |w_1| \\ \text{ED}(nil, w_2) &= |w_2| \\ \text{ED}(nil, nil) &= 0 \end{aligned}$$

The edit distance for a pair of words is efficiently computed via *dynamic programming*, If needed, Gusfield [45] (pp. 215-223) provides a good explanation of how dynamic programming is applied to compute the edit distance. Since dynamic programming is generally taught in graduate-level computer science courses, I will omit the details here.

Both LCSR and EdSim yield real values ranging from 0 to 1. Very dissimilar words will have zero or close to zero LCSR and EdSim values, while exactly identical words will have LCSR and EdSim equal to 1. Since these measures do not change their behaviour depending on the languages under consideration, we call them *static*.

5.2.2 Adaptable measures

Several language-adaptable measures have been proposed earlier [10, 57, 95, 106], but there are no readily available implementations of any of these measures, which prevents reproducibility of the reported results. Also, because each of these measures has been evaluated on a different language pair and using datasets collected in different ways, the reported results cannot be meaningfully compared to each other.

Cognate Alignment via Cohesive Character Sequences

The technique for cognate alignment proposed by Ribeiro et al [95] is based on contiguous and non-contiguous cohesive character sequences that are common to the two languages under consideration. For example, the non-contiguous sequence "li_re" matches both "livre" and "libre" in Portuguese and French, respectively. These cohesive sequences are first extracted using a statistic method (SENTA) proposed by Dias [33] from a corpus containing texts of both languages. Then, taking a pair of parallel documents as the alignment scope, occurrences of previously extracted sequences in both documents are paired together, giving rise to alignment candidates. Cohesive sequences may include the whitespace character, represented as "·", and thus occurrences of these sequences may span over

multiple words. Also, occurrences of different sequences may overlap each other and they will be merged together, possibly forming a longer segment. For example the sequences "·_eclar", "clara" and "lara__o" are all matched within the strings "·declaration" and "·declaração" (note the whitespace at the beginning). Alignment candidates are then statistically filtered based on their positions in the documents.

Interestingly, this method is not only able to align words but also phrases, such as "libre circulation"↔"livre circulação" (French and Portuguese for "free movement").

Since this method was proposed as an alignment method and not an extraction method, it is understandable that the authors did not evaluate its performance in terms of precision and recall of the aligned words or phrases, as would be expected for an extraction method.

Co-occurrence of Short Character Sequences

Tiedemann [106] proposed a method[2], called NMmap, to automatically create adapted similarity measures based on co-occurrence association statistics of short character sequences within cognates.

The NMmap method is based on character-wise string alignments, such as the one depicted in Figure 5.1. This alignment is obtained via dynamic programming, in a similar way that edit distance is computed. Once again, we refer to Gusfield [45] (pp. 215-223) for a good explanation of how dynamic programming is applied to compute character-wise string alignments, if needed.

t	e	l	e	ph	o	n	e
t	e	l	e	f	o	n	e

Figure 5.1: Character-wise alignment of "telephone" and "telefone" (Portuguese).

From a pair of aligned words we extract all pairs of mismatched segments with at most three characters, such as "ph" and "f" in Figure 5.1. Each pair of mismatched segments should be assigned a weight based on its frequency within a set of cognates known apriori of the language pair under consideration. The paper gives a hint that the similarity score for a pair of words is to be somehow derived from the weights of the mismatched segments found in them, but the equation for computing the score is not given.

Tiedemann compared NMmap against LCSR in a task of cognate identification for the Swedish-English language pair and observed a gain of +3% in terms of estimated precision and 21% in terms of recall.

[2] Actually, the paper presents three methods, but the first two methods are reported to have 13% and 61% lower recall than LCSR for similar precision values. Given that these methods are much more complex than LCSR and yet perform worse, we do not include them in our overview.

Using an SVM Classifier to Identify Cognates

Bergsma and Kondrak [10] propose the use of a SVM classifier [22] to identify cognates. The features given as input to the classifier include static similarity measures, such as LCSR, and a large number of binary features which take value 1 when specific pairs of character sequences co-occur in aligned positions of the words and 0 otherwise.

Their method starts by aligning words character-wise, as in the method proposed by Tiedemann [106], described above. From aligned words, they extract pairs of aligned sequences of characters with lengths varying between 1 and 3 characters.

For example, from the words "telephone" and "telefone" aligned in Figure 5.1, they would extract the following pairs of aligned sequences:

"t"↔"t", "te"↔"te", "tel"↔"tel", "e"↔"e", "el"↔"el", "ele"↔"ele", "l"↔"l", "le"↔"le", "eph"↔"ef", "ph"↔"f", "pho"↔"fo", "o"↔"o", "on"↔"on", "one"↔"one", "n"↔"n", "ne"↔"ne"

According to Bergsma and Kondrak, the extraction of these sequences was inspired by the algorithm proposed by Koehn [56] for extracting aligned phrases from word alignments, which was briefly described in the previous chapter. The methods are similar, except that, instead of aligned words, here we deal with aligned characters.

Besides these character sequences with length lower than or equal to 3, Bergsma and Kondrak also extract longer sequences for mismatched segments, starting one character before and ending one charachter after the mismatched segment.

In this example, there would be only one such segment: "epho"↔"efo".

The main idea of this method is that the SVM will be able to learn that some aligned character sequences are indicators of equivalence while others indicate the opposite.

5.3 Proposed Spelling Similarity Measure: SpSim

The SpSim measure was inspired by the dynamic methods proposed by Tiedemann [106] and Bergsma and Kondrak [10]. Like in these two methods, I start by aligning words character wise. However, instead of considering a large number of arbitrary character sequences, like these methods do, I focused on the non-identical sequences that occupy the same position in cognates, such as "ph"↔"f", which appears in "pharmacy" into "farmácia", "photographic" and "fotográfica", "phase" and "fase" and many other cognates. I call these mismatched pairs of sequences *substitutions*.

By hypothesis, each language pair has a specific set of recurrent substitutions which we can extract from example cognates.

The first step for computing the proposed spelling similarity (SpSim) measure [43] is to extract all substitutions from a given pair of words. Then, like the EdSim and LCSR measures, we compute the distance as the ratio of characters within mismatched segments over the total number of characters of the longest word. However, unlike those measures,

we will ignore segments where the words differ **but** match a known substitution pattern of the language pair under consideration.

The next subsection formalizes the concept of substitution pattern, and shows how we extract them from known cognates. Then, we present the SpSim equation to compute the spelling similarity between words, and we show how it relates to EdSim.

5.3.1 Bilingual Substitution Patterns

SpSim relies on standard character-based string alignment using dynamic programming. A detailed explanation of string alignment algorithms is given by Gusfield [45]. We compute the best global alignment of the two strings using the standard weighting scheme (uniform weights for the three possible character-based operations: insertion, deletion, and replacement).

To aid the explanation we will use two English-Portuguese cognate word pairs as an example: "photographic" and "fotográfica", and "achromatic" and "acromático". The alignment of these cognates, is shown below:

```
photographic       achromatic
::|||||:::||:      ||:|||:|||:
 fotográf ica      ac romático
```

In this representation the pipe ("`|`") character marks aligned matches, while the colon ("`:`") character marks aligned mismatches. If we eliminate all matched characters, we are left with the mismatched segments that contain some recurrent substitutions like ("`ph`", "`f`"):

```
ph     aph           h   a
::     :::  :        :   :   :
 f     áf   a            á   o
```

However, because we removed the context around the mismatched segments, some of these patterns are too generic: the insertion of "a", the deletion of "h" and the insertion of "o", are patterns that are likely to appear even if the words are totally unrelated.

To improve the specificity of the patterns, we include a fixed context window of characters around each pattern. Empirically, we settled on a window of one character, as it provides a good balance between specificity and generality. Furthermore, we introduced two special characters, the caret (^) and the dollar sign ($) at the beginning and end of the aligned strings, respectively. These two characters are not allowed to appear in the words, and they allow us to distinguish patterns that appear in the beginning, middle or end of words. The patterns extracted with context window of length 1 are presented below:

```
^pho   raphi c $    chr mat c $
|::|   |:::| |:|    |:| |:| |:|
^ fo   rá fi ca$    c r mát co$
```

Now we have patterns much more specific, such as: the insertion of an "a" at the end of a word whose last character is a "c". The complete list of patterns extracted from these two words is: ("^pho", "^fo"), ("raphi", "ráfi"), ("c$", "ca$"), ("chr", "cr"), ("mat", "mát"), and ("c$", "co$").

5.3.2 Generalization of substitution pattern contexts

The context characters of these patterns may be dropped if we find the same substitution in a different context, making the pattern more general, i.e. applicable in more contexts. For example, applying the method given in the previous subsection to "phase" and "fase" we extract the pattern ("^pha", "^fa"), containing the substitution ("ph", "f") that is also found in the pattern ("^pho", "^fo"), extracted from the previous examples.

The generalization is done by dropping the context characters that are different between patterns containing the same substitution. In this case, we drop the right context character from both patterns and we obtain the generalized pattern ("^ph", "^f").

5.3.3 SpSim equation

The equation for computing SpSim is:

$$SpSim(w_1, w_2) = 1.0 - \frac{D(w_1, w_2)}{\max(|w_1|, |w_2|)} \tag{5.3}$$

Where w_1 and w_2 are the words being compared, $D(w_1, w_2)$ is a distance measure that returns the length[3] of all mismatched segments containing unknown substitutions, $|w_1|$ is the length of w_1, and $|w_2|$ is the length of w_2.

As the number of extracted substitution patterns increases, $D(w_1, w_2)$ becomes lower for pairs of cognates where those patterns appear. More formally, the distance measure is defined as:

$$D(w_1, w_2) = ED(w_1, w_2) - \sum |m_i|, \forall m_i \in \mathcal{M}(w_1, w_2) \cap \mathcal{K} \tag{5.4}$$

Here, $ED(w_1, w_2)$ is the edit distance (the total number of mismatched characters on both strings), $\mathcal{M}(w_1, w_2)$ is the set of mismatched segments between the two strings, and $\mathcal{K}$ is the set of known substitutions. To understand the $D(w_1, w_2)$ distance measure, lets start by considering the situation where $\mathcal{M}(w_1, w_2) \cap \mathcal{K}$ is an empty set. That happens when all mismatched segments ($\mathcal{M}$) are unknown substitutions, i.e., none of them belongs to the set of known substitutions ($\mathcal{K}$). In this situation, the value of $D(w_1, w_2)$ is exactly the same as $ED(w_1, w_2)$. Therefore, without having extracted substitutions from examples, SpSim is mathematically equivalent to EdSim.

Conversely, whenever $M(w_1, w_2) \cap \mathcal{K}$ is non empty, it means that at least some mismatched segments are substitutions that were previously extracted. Each $|m_i|$ is the length of the mismatched segment. It is easy to see that the inequation

[3] All lengths are in terms of characters.

English	Portuguese	EdSim	LCSR	SpSim	Known Diffs
recommended	recomendada	0.72	0.82	1.00	mm-m, e-a, d$-da$
decembers	dezembros	0.66	0.78	1.00	c-z, er-ro
efficiency	eficiência	0.60	0.70	1.00	ff-f, e-ê, y$-ia$
masters	mestres	0.57	0.71	1.00	a-e, er-re
memory	memória	0.57	0.57	1.00	o-ó, y$-ia$
used	usados	0.50	0.50	1.00	e-a, d$-dos$
both	ambas	0.00	0.20	0.00	

Table 5.1: Example EdSim, LCSR and SpSim scores for several translation equivalents. Known spelling differences found on each pair of words are listed in the rightmost column.

$$D(w_1, w_2) \leq ED(w_1, w_2) \tag{5.5}$$

is always true, which means that SpSim will always assign similarity values higher or equal than EdSim for any given pair of strings. In the limit, if all mismatched are known substitutions, then the sum of all $|m_i|$ is equal to $ED(w_1, w_2)$, and $SpSim(w_1, w_2)$ is 1.0. Thus, when SpSim yields 1.0 for a given pair of strings, it does not imply that *the strings are identical*, but instead, that *all mismatched segments in the strings are known substitutions of the language pair under consideration*.

5.3.4 Examples

Next, let us consider that we have a hashtable containing the patterns from the examples given earlier in Sub-Section 5.3.1: (“^pho”, “^fo”), (“raphi”, “ráfi”), (“c$”, “ca$”), (“chr”, “cr”), (“mat”, “mát”), and (“c$”, “co$”).

To compute the SpSim of the words “phonetic” and “fonético” we align them and we extract the substitution patterns as described earlier:

```
^pho net c $
|::| |:| |:|
^ fo nét co$
```

Then, we look up each substitution in the hashtable and we sum the $|m_i|$ values. In this case, the hashtable contains the fist and last patterns, which have $|m_i|$ values of 2 and 1, respectively.

The edit distance of the two words is easily obtained by counting the number of mismatched characters in the alignment, which is 4. The maximum lenght of the two words is 8 characters. Thus, substituting all values in equation 5.3 we obtain a SpSim measure of 0.875. By contrast, the EdSim measure for these two words is only 0.5.

Table presents some translation equivalents and the respective EdSim, LCSR and SpSim scores. SpSim extracted and memorized substitution patterns from a large list of examples, described later in Section 5.4.

From Table 5.1 we observe that SpSim assigns a score of 1.0 very often. As discussed in Sub-Section 5.3.3, a 1.0 score does not imply that the strings are identical, but instead that all the differences found in a pair of words are known. Nevertheless, as expected, SpSim does not give high scores to all pairs of words, as we can see in the last example of the table. The words "both" and "ambas", despite being translation equivalent, and both of them having a "b", do not align at any single character.

The shortest and thus less costly character-wise alignment of both strings is given below at the left:

```
^both $       ^  both$
|:::::|       |::|:::|
^ambas$       ^ambas $
```

The alignment on the right, while might look more intuitive, has the same number of mismatched characters (5).

5.3.5 Operation Modes

There are two fundamental operation modes:

In *adaptation mode* we give pairs of known cognates as input, which we call *examples*, and from each example we extract new substitution patterns or we generalize previously extracted ones, depending on if the mismatched segment of the pattern is already known or not.

In *evaluation mode* we use the set of substitution patterns previously extracted while operating in adaptation mode, to compute the spelling similarity of new pairs of words given as input.

In terms of hashtable operations, for each mismatched segment found in a pair of words, we will do one query to the hash table. When in adaptation mode, the query may be followed by an insertion or an update of the associated context, depending on if the mismatched segment of the pattern is already known or not and wether the context was generalized or not.

The asymptotic time cost of the hashtable insert and query operations is constant, and thus the overall time cost of SpSim is dominated by the dynamic programming string alignment. Since the alignment procedure is common to SpSim, EdSim and LCSR, it means that the three measures have the same asymptotic time cost.

5.4 Evaluating Similarity Measures

Since there are no readily available implementations of previously proposed dynamic methods, and given they are too complex to reproduce from the succinct descriptions given in the respective papers, direct comparison with those methods is not possible. Therefore, this section presents the results of an experiment carried out over 4 language

pairs, to compare the performance of SpSim, EdSim and LCSR in the task of extracting bilingual pairs of equivalent cognates, or just cognates hereafter.

5.4.1 Source Parallel Corpus

As the source parallel corpus for extracting cognates we used the English, French, German, Portuguese and Spanish translations of the Treaty establishing a Constitution for Europe, which is an unratified treaty that was intended to create a consolidated constitution for the European Union [4]. This treaty was selected for three reasons: first, because it is big enough for co-occurrence statistics to be meaningful, second, because it is translated in many languages, and third, because the text was carefully and closely translated, which facilitates sentence-level alignment across all languages and provides a clean base for the rest of the experiment.

The documents were first sentence-aligned, for each language pair, using the method described in Chapter 3, and afterwards the alignments were manually corrected as needed. Some noisier and less interesting parts of the documents were discarded, such as annexes.

The number of tokens in the clean sub-corpus of each language is given below.

English (EN)	German (DE)	Spanish (ES)	French (FR)	Portuguese (PT)
17415	16704	18750	19908	18726

5.4.2 List of Candidate Translation Equivalents

From each language pair of the sentence-aligned corpus described above, we will extract a list of pairs of words with high co-occurrence similarity.

In his PhD thesis, Ribeiro evaluated 28 co-occurrence similarity measures [94] for the purpose of extracting translation equivalents and concluded that the Dice coefficient was one of the most accurate measures. Because the Dice coefficient is also one of the simplest measures, and thus easier to understand, we adopted it as an initial filter to produce a list of candidate word pairs based on their co-occurrence.

For a pair of words (w_1, w_2), the Dice coefficient is given by the following equation:

$$Dice(w_1, w_2) = \frac{2 * F(w_1, w_2)}{F(w_1) + F(w_2)} \tag{5.6}$$

Where $F(w_1)$ is the frequency of w_1 in the sentences of language L_1 and $F(w_2)$ is the frequency of w_2 in the sentences of language L_2; $F(w_1, w_2)$ is the *co-occurrence frequency* of the words in the sentence-aligned parallel corpus, i.e. the number of times that w_1 and w_2 occur in aligned sentences. Thus, the Dice coefficient will be 1, if word w_2 occurs in all sentences aligned with sentences where w_1 occurs, and vice versa. The Dice coefficient will be 0, if the words do not co-occur.

[4] The documents were retrieved from `http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=OJ:C:2004:310:TOC`.

As said above, for each language pair of the sentence-aligned corpus, we will extract a list of pairs of words with high co-occurrence similarity. Thus, we extracted all pairs of words that co-occur in at least two aligned sentences and having *Dice coefficient* greater than or equal to 0.6. Identical words were excluded from this list since, by definition, they have maximum spelling similarity and will not help us comparing SpSim, LCSR and EdSim.

All word pairs in these lists were labeled by a linguist as being TE or not, independently of being cognates or not.

The percentage of extracted equivalent and not equivalent word pairs for each language pair are shown in Table 5.2.

	EN-DE	EN-ES	EN-FR	EN-PT
Extracted	1147	1148	1205	1206
Equivalent	23.5%	34.8%	31.5%	34.0%
Not Equivalent	76.5%	65.2%	68.5%	66.0%

Table 5.2: Results of the extraction by thresholding Dice above 0.6 for each language pair. The last row shows the percentage of word pairs that were considered correct translations by linguists.

We will hereafter refer to these lists of word pairs as our *testsets*.

5.4.3 List of Example Translation Equivalents

For each of the four language pairs, we retrieved a list of all word-word translations in our human-validated lexicon. From this list we removed all word pairs that are part of the testset. We refer to the resulting lists as *examples* lists and their sizes are presented in Table 5.3. SpSim was given these lists for extracting and memorizing substitution patterns for each language pair separately. The English-Portuguese list of examples is much larger than the others because this language pair has been worked on for longer.

	EN-DE	EN-ES	EN-FR	EN-PT
# Examples	68259	108129	139241	224279

Table 5.3: Number of examples given to SpSim for each language pair.

5.4.4 Evaluating and Comparing Spelling Similarity Scores

Each word pair from the testset was given to SpSim, LCSR and EdSim, for computing the respective spelling similarity scores.

We evaluated the performance of SpSim, LCSR and EdSim in terms of precision, recall and combined F-measure by selecting a subset of word pairs from the testset with spelling similarity score above a specific threshold.

Precision was computed as the percentage of true equivalents extracted over the total number of word pairs extracted. Recall was computed as the percentage of true equivalents extracted over the total number of true equivalents in the testset. The F-measure was computed as the harmonic mean between precision and recall.

Table 5.4 presents the precision (P), recall (R) and F-measure (F) for 11 thresholds ranging from 0 to 1 by steps of 0.1. This table includes the results for EdSim, LCSR, and SpSim side by side and for each of the four language pairs. The highest F-measure for each similarity measure and language pair appears in **boldface**.

As expected, increasing the threshold tends to increase the precision of all measures, but the recall decays. However, we see that recall of EdSim and LCSR decays much more than the recall of SpSim. The decay of recall as we increase the threshold is easier to see in the plots presented in Figure 5.2. Note how the recall of SpSim remains high as the threshold is increased. If we select a threshold for each measure such that its precision is about 90%, then SpSim would have a much higher recall than EdSim and LCSR.

Table 5.5 presents a cleaner view of the most important information in Table 5.4, with only the optimal thresholds of each measure for each language pair. The optimal threshold was selected as the one that maximizes the F-measure. In this table it is easier to see that the performance of EdSim is slightly better than LCSR for all language pairs except English-French and that both static measures have peak F-measure with much lower precision than SpSim (0.9 to 1.0). The lower thresholds of EdSim and LCSR have a direct impact in the precision of these methods, which is also significantly lower than SpSim.

Table 5.6 further distills the experimental results into comparative deltas indicating the difference between SpSim precision, recall and F-measure scores and the corresponding scores of the static measure with highest F-measure for each language pair. SpSim yields the highest F-measure for all language pairs by a margin ranging from 9.1% to 21.4%. Furthermore, the highest F-measure is attained by SpSim at the same thresholds that it obtains the highest precision. This means that, not only the precision is the highest, but also that the precision and recall are balanced at that point.

5.5 Summary

In this chapter, we propose a new measure for spelling similarity that dynamically adapts to recognize and ignore spelling differences that are recurrent in the pair of languages under consideration. Since there are no implementations readily available for other dynamic measures proposed earlier, a direct comparison was not possible.

The comparative evaluation of SpSim against two baseline static measures, EdSim and LCSR, shows that SpSim outperforms these measures, for all 4 language pairs considered (English-Spanish, English-French, English-Portuguese, English-German, and French-Italian). For the same threshold values, SpSim has much higher recall than the baseline measures while achieving a similar precision. As a consequence, the optimal

		EdSim			LCSR			SpSim		
L1-L2	T	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
EN-DE	0.0	23.5	100.0	38.0	23.5	100.0	38.0	23.5	100.0	38.0
	0.1	28.0	77.7	41.1	26.2	97.0	41.3	29.0	90.7	44.0
	0.2	41.6	51.7	46.1	31.8	83.6	46.1	34.6	85.1	49.2
	0.3	65.4	39.4	**49.2**	43.5	50.2	46.6	40.4	79.6	53.6
	0.4	86.4	33.1	47.8	63.2	38.3	**47.7**	47.8	72.9	57.7
	0.5	92.6	27.9	42.9	81.3	32.3	46.3	57.1	68.4	62.3
	0.6	95.2	21.9	35.6	95.5	23.4	37.6	70.9	58.7	64.2
	0.7	97.8	16.4	28.0	98.0	18.2	30.7	81.0	56.9	66.8
	0.8	100.0	9.3	17.0	100.0	10.0	18.2	89.3	55.8	68.6
	0.9	100.0	1.5	2.9	100.0	1.5	2.9	90.3	55.4	68.7
	1.0	N.A.	0.0	N.A.	N.A.	0.0	N.A.	90.9	55.4	**68.8**
EN-ES	0.0	34.8	100.0	51.6	34.8	100.0	51.6	34.8	100.0	51.6
	0.1	44.2	91.5	59.6	36.9	96.7	53.5	42.3	93.7	58.3
	0.2	62.6	79.2	69.9	45.4	91.5	60.7	48.6	91.5	63.5
	0.3	80.8	71.7	**76.0**	67.2	79.2	72.7	52.8	89.5	66.4
	0.4	91.1	63.9	75.1	80.7	71.4	**75.8**	60.8	87.2	71.7
	0.5	95.3	61.2	74.5	90.9	64.9	75.7	67.5	85.5	75.4
	0.6	98.0	48.4	64.8	97.2	51.6	67.4	77.8	81.0	79.4
	0.7	98.1	39.8	56.7	98.3	42.9	59.7	86.0	77.2	81.4
	0.8	100.0	25.8	41.0	100.0	28.8	44.7	92.3	75.4	83.0
	0.9	100.0	4.5	8.6	100.0	5.3	10.0	94.6	74.9	**83.6**
	1.0	N.A.	0.0	N.A.	N.A.	0.0	N.A.	94.6	74.9	83.6
EN-FR	0.0	31.5	100.0	47.9	31.5	100.0	47.9	31.5	100.0	47.9
	0.1	39.9	89.2	55.1	33.2	95.5	49.3	37.4	91.8	53.2
	0.2	58.0	76.1	65.8	40.6	90.3	56.0	42.9	87.1	57.5
	0.3	74.5	68.4	**71.3**	63.2	78.7	70.1	48.1	85.5	61.6
	0.4	84.5	61.6	71.2	75.2	69.5	**72.2**	54.1	82.4	65.3
	0.5	89.8	57.9	70.4	85.6	61.1	71.3	60.5	79.5	68.7
	0.6	91.6	46.1	61.3	90.8	49.5	64.1	72.2	75.8	73.9
	0.7	93.4	37.4	53.4	93.4	40.8	56.8	78.6	72.6	75.5
	0.8	92.1	24.5	38.7	92.7	26.8	41.6	84.0	71.8	77.4
	0.9	88.6	8.2	14.9	89.2	8.7	15.8	86.9	71.6	**78.5**
	1.0	N.A.	0.0	N.A.	N.A.	0.0	N.A.	86.9	71.6	78.5
EN-PT	0.0	34.0	100.0	50.7	34.0	100.0	50.7	34.0	100.0	50.7
	0.1	44.9	92.2	60.4	36.6	98.3	53.4	41.8	95.4	58.1
	0.2	63.9	80.0	71.1	45.4	92.0	60.8	47.2	93.2	62.7
	0.3	82.4	72.9	**77.4**	67.4	77.6	72.1	51.5	92.2	66.1
	0.4	88.4	65.1	75.0	81.3	71.2	**75.9**	58.2	89.8	70.6
	0.5	91.4	60.0	72.5	89.1	63.9	74.4	64.7	88.0	74.6
	0.6	94.2	47.3	63.0	92.0	50.2	65.0	76.0	82.4	79.1
	0.7	95.1	28.3	43.6	94.9	32.0	47.8	85.1	80.7	82.9
	0.8	93.4	13.9	24.2	94.5	16.8	28.6	88.6	79.8	84.0
	0.9	100.0	3.9	7.5	95.2	4.9	9.3	90.3	79.8	**84.7**
	1.0	N.A.	0.0	N.A.	N.A.	0.0	N.A.	90.3	79.8	84.7

Table 5.4: Precision (P), recall (R) and F-measure (F) for thresholds (T) ranging from 0 to 1 for each of the 4 language pairs, for EdSim, LCSR and SpSim. The highest F-measure values for each similarity measure and language pair combination appear in bold.

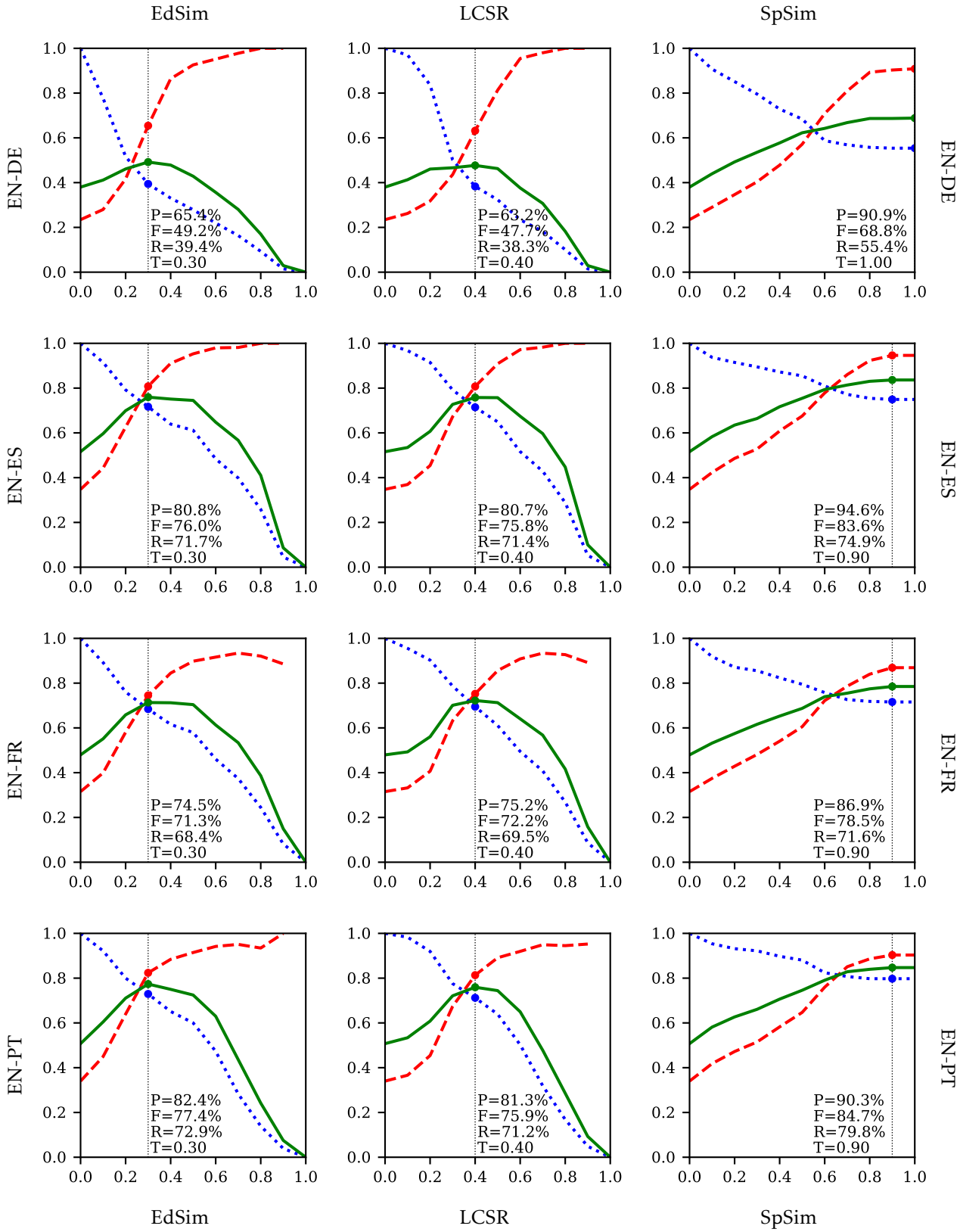


Figure 5.2: Plots of precision (P – – –), recall (R ……) and F-measure (F——) for thresholds (T) ranging from 0 to 1 for each of the 4 language pairs, for EdSim, LCSR and SpSim. The highest F-measure value in each plot is shown together with threshold, precision and recall at that point.

	EdSim				LCSR				SpSim			
L1-L2	T	P	R	F	T	P	R	F	T	P	R	F
EN-DE	0.3	65.4	39.4	49.2	0.4	63.2	38.3	47.7	1.0	90.9	55.4	68.8
EN-ES	0.3	80.8	71.7	76.0	0.4	80.7	71.4	75.8	0.9	94.6	74.9	83.6
EN-FR	0.3	74.5	68.4	71.3	0.4	75.2	69.5	72.2	0.9	86.9	71.6	78.5
EN-PT	0.3	82.4	72.9	77.4	0.4	81.3	71.2	75.9	0.9	90.3	79.8	84.7

Table 5.5: Precision (P), Recall (R) and F-measure (F) of EdSim, LCSR and SpSim at the threshold (T) providing maximum F-measure for each language pair.

	SpSim – (2nd best)		
L1-L2	ΔP(%)	ΔR(%)	ΔF(%)
EN-DE	+25.5	+16.0	+19.6
EN-ES	+13.8	+3.2	+7.6
EN-FR	+11.7	+2.1	+6.3
EN-PT	+7.9	+6.9	+7.3
average	+14.7	+7.1	+10.2

Table 5.6: Absolute performance improvements of SpSim over EdSim and LCSR. The delta values for precision (ΔP), recall (ΔR) and F-measure (ΔF) are the difference of the corresponding scores obtained by SpSim and the static measure with highest F-score for each language pair.

threshold for SpSim (the one that enables the maximum F-measure) also yields much higher precision (from 97.1% to 99.7%) than the baseline measures (from 63.8% to 93.4%), which is desirable in a scenario where the extracted cognates are to be human validated.

Despite its much better performance, the SpSim algorithm still has the same asymptotic time complexity as the EdSim and LCSR static measures, and is quite simple to implement (the reference implementation has 80 lines of Python code, including comments).

The work presented in this chapter has already been published [43] and the source code is available from github[5] under an open-source license. Furthermore, the paper has been cited by other authors and the findings reported here have been independently confirmed. For example, Ciobanu and Dinu [21] wrote:

"Out of the four similarity metrics, SpSim obtains, overall, the best performance. These results support the relevance of accounting for orthographic cues in cognate identification."

SpSim is language independent and only requires a list of manually validated cognates from which the substitution patterns are extracted automatically. Therefore, this method satisfies the main underlying goal of this thesis of reusing previously validated knowledge to acquire more knowledge (new bilingual cognate pairs).

[5]the source of SpSim is available from `https://github.com/luismsgomes/spsim`

Chapter Six

Pattern-Based Extraction

> When I look at an article in Russian, I say: "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."
>
> Warren Weaver, 1947, quoted in John Hutchins [47], p. 195

6.1 Introduction

I got the following insightful observation from my supervisor after I presented him the work of the previous chapter: What if instead of the equivalence between "ph" and "f" in English-Portuguese cognates we knew that the stems "learn$*$" and "aprend$*$" are equivalent as well as the suffixes "$*$ed" and "$*$eu"? Then, we could infer that words "learned" and "aprendeu" are equivalent, despite not being cognates.

The suggested possibility of deducing equivalence between words from equivalence of stems and suffixes, motivated me to develop a more general method to extract translation equivalents (TEs). A method that does not rely on the cognaticity of words and thus is applicable to words even if their spelling is quite different.

Besides the possibility of matching equivalent stems and suffixes, we found many other regularities in translation equivalents that can be exploited, particularly when dealing with multi-word expressions. For example, the TEs "regarding"↔"que respeita", "establishing"↔"que estabelece", and "being"↔"que sejam" follow a common pattern. Which pattern is that? In plain text, we could describe the pattern as follows:

- all these TEs consist of a single English word translated as two Portuguese words;
- the English word ends with suffix "$*$ing";
- the Portuguese phrase begins with "que";
- the stem of the English word is equivalent to the stem of the second Portuguese word.

Is there a more compact and easier way to express such pattern?

As another example, consider the English-Portuguese translations "workplace"↔"local de trabalho", "firearm"↔"arma de fogo" or "database"↔"base de dados". We immediately recognize a common pattern across these three equivalents, despite their quite different meanings. How can we express this pattern?

This chapter has two goals. The first is to provide a language to allow us express a wide range of translation patterns in a concise but human-readable form. The second is to provide a method for matching these patterns within parallel sentences, thus enabling the extraction of new translation equivalents.

6.2 Proposed Language for Expressing Translation Patterns

In this section I will propose a language for expressing translation patterns in a concise but human readable form, which is the first goal of this chapter.

Translation patterns are rules that follow a specific syntax. In each of the following sub-sections we gradually introduce syntax and concepts of translation patterns.

6.2.1 Variables and Suffix Restrictions

Let us start by considering the Czech-Portuguese verb translations "pokračovat"↔"continuar" (continue), "pracovat"↔"trabalhar" (work), "tancovat"↔"dançar" (dance).

All these word pairs share the suffix pair "∗ovat"↔"∗ar" and, even if we do not know Czech or Portuguese, we can assume that the remainder of the words, i.e. their stems, should be equivalent.

Thus, to match word translations like these, we want a translation pattern that matches pairs of words with equivalent stems and these specific suffixes. For this, I propose the syntax "$1/ovat"↔"$1/ar", where $1 is a *linked variable* that represents a stem and its translation. We say that the variables in this pattern are *morphologically restricted* because we specified word suffixes.

6.2.2 Matching Multi-word Expressions

Let us now look at patterns that match multi-word expressions. Consider, for example, the English-Portuguese noun-phrase translations presented in Table 6.1. We used angle brackets with subscript numbers to identify corresponding words or phrases within each TE.

The numbered brackets make clear that all these TEs follow a common reordering pattern and that the Portuguese preposition "de" does not have a direct equivalent word in the English translations.

The syntax that expresses this pattern is thus "$1 $2"↔"$2 de $1".

Compared to the pattern presented earlier, this pattern introduces three novelties: (1) it contains multiple variables ($1 and $2); (2) there is a literal "de" in the right-hand side

$$\langle \text{swap} \rangle_1 \ \langle \text{contract} \rangle_2 \leftrightarrow \langle \text{contrato} \rangle_2 \text{ de } \langle \text{swap} \rangle_1$$
$$\langle \text{market} \rangle_1 \ \langle \text{practice} \rangle_2 \leftrightarrow \langle \text{práticas} \rangle_2 \text{ de } \langle \text{mercado} \rangle_1$$
$$\langle \text{performance} \rangle_1 \ \langle \text{management} \rangle_2 \leftrightarrow \langle \text{gestão} \rangle_2 \text{ de } \langle \text{desempenho} \rangle_1$$
$$\langle \text{press} \rangle_1 \ \langle \text{office} \rangle_2 \leftrightarrow \langle \text{gabinete} \rangle_2 \text{ de } \langle \text{imprensa} \rangle_1$$
$$\langle \text{preliminary planning} \rangle_1 \ \langle \text{phase} \rangle_2 \leftrightarrow \langle \text{fase} \rangle_2 \text{ de } \langle \text{planeamento preliminar} \rangle_1$$
$$\langle \text{open market} \rangle_1 \ \langle \text{operation} \rangle_2 \leftrightarrow \langle \text{operação} \rangle_2 \text{ de } \langle \text{mercado aberto} \rangle_1$$

Table 6.1: Example English-Portuguese noun phrase translations matched by multi-variable pattern "$1 $2"↔"$2 de $1".

of the pattern; and (3) the variables in this pattern are not morphologically restricted. Let us look at each of these new features in more detail:

1. Multiple variables allow us to express multi-word structures. They are of particular interest for capturing local word reorderings within phrasal translations, as demonstrated in the examples above. By convention, variables are numbered sequentially on the left-hand side, starting at 1. Thus, a word reordering will be expressed in a non-increasing sequence of variables on the right-hand side.

2. Literal words or phrases in the patterns, such as the preposition "de" in the translation pattern "$1 $2"↔"$2 de $1", allow us to write patterns for phrases containing words that have no direct equivalent in the other language. In the examples of Table 6.1, the function of the Portuguese preposition "de" is carried out by the reversed word ordering in the English side, but it cannot be independently aligned to any particular word. This is a situation where the strictly word-based alignment and translation models, discussed earlier in Chapter 4, show their inadequacy, given that they focus on the individual words and have very weak (or inexistent) models for the contexts where they appear.

3. Morphologically-unrestricted variables such as $1 are more general than the restricted ones we saw earlier, obviously. But the increase in generality does not come solely from the absence of morphological restrictions. It comes from the fact that morphologically-unrestricted variables are not limited to match single words, but instead are allowed to match phrases as well. This was the case in the last two examples of Table 6.1, where $1 was instantiated by multi-word noun phrases: "preliminary planning"↔"planeamento preliminar" and "open market"↔"mercado aberto". Note that, like morphologically-restricted variables, we require stem-level equivalence of phrases matching the variables, as described earlier in Subsection 4.3.7. If the matching of phrases is not desired for a specific variable, there is a way to force the variable to be matched only by words. It is enough to use the same syntax that we would use for specifying a suffix, but we leave the suffix *empty*, as for example "$1/".

Returning to our discussion of the pattern "$1 $2"↔"$2 de $1", although we are not imposing any morphological restrictions to these variables, this pattern is still relatively precise, in terms of what it matches: we are requiring that a pair of consecutive English words or phrases ($1 $2) are equivalent at a stem level to words or phrases that appear in the Portuguese sentence in the reverse order and have the preposition "de" between them ($2 de $1). In the experiments that will be described later, this simple pattern enabled the extraction of 3314 phrase equivalents, such as the ones presented in Table 6.1, with a precision of 99%. Note that most of these phrases occur only once or twice in the corpus. Typically, lexica extraction methods based on statistics do not even attempt to extract translations for words or phrases with such low frequency due to lack of statistical significance of such rare events. Furthermore, even in recent extraction methods based on *deep* syntatic and semantic analysis, phrases with only one occurrence are filtered out [102].

6.2.3 Matching Compound Words

Matching of compound words is very important for agglutinative languages such as German, but not only.

For example, consider the English-Portuguese TEs "workplace"↔"local de trabalho", "firearm"↔"arma de fogo" or "database"↔"base de dados" presented earlier in the introduction.

To match these TEs we need a translation pattern that matches two distinct pairs of equivalent stems, and thus should have two variables, but the stems are combined in a single English word.

This is expressed intuitivelly, I hope, by writing variables together, in the same way that compounds are formed by concatenating word stems. Thus, a pattern for matching the given examples would be "$1$2"↔"$2 de $1".

6.2.4 Prefix Restrictions

In the same way that we can restrict variables to match words with specific suffixes, we can also restrict variables based on word prefixes. To specify a prefix we prepend it to the variable, separating it with a backslash (\) as in the example pattern "in\$1"↔"não $1". Table 6.2 shows some phrase pairs matched by this pattern. Because a word prefix is conceptually symmetric to a word suffix, we use the backslash (\) to separate prefixes from the variable and the forward slash (/) to separate suffixes. For the same reason, prefixes and suffixes are specified before and after the variable, respectively.

6.2.5 Lists of Alternative Prefixes and Suffixes

In some situations, it is useful to restrict variables based on a set of alternative word suffixes or prefixes, instead of a single one. Of course, we could write several patterns,

inedible	↔ não comestíveis
inadmissible	↔ não admissíveis
insignificant	↔ não significativo
informal	↔ não formal
insoluble	↔ não solúveis

Table 6.2: Example English-Portuguese phrase translations matched by prefix-restricted pattern "in\$1"↔"não $1".

one for each suffix or prefix, but it is easier to have a syntax that allows us to specify multiple suffixes or prefixes in the same pattern. Therefore, we write multiple prefixes or suffixes next to each other, separating them with a back slash (\), in the case of prefixes, or a forward slash (/), in the case of suffixes. For example, the first six patterns on Table 6.3 differ only in the allowed suffixes on the Portuguese side. The last pattern (seventh) is equivalent to the first six and matches all the example phrase pairs listed on the right column.

For example, the prefix-restricted pattern "in\$1"↔"não $1" could be extended to include prefixes un\ and im\, as "in\im\un\$1"↔"não $1". Then, it would match "unsealed"↔"não selado" and "imperishable"↔"não perecível" as well as the phrase pairs that we saw earlier in Table 6.2.

Pattern	Example Phrase Pair
is to be $1/ed ↔ se deve $1/ar	is to be classified ↔ se deve classificar
is to be $1/ed ↔ se deve $1/er	is to be maintained ↔ se deve manter
is to be $1/ed ↔ se deve $1/ir	is to be required ↔ se deve exigir
is to be $1/en ↔ se deve $1/ar	is to be taken ↔ se deve retirar
is to be $1/en ↔ se deve $1/er	is to be written ↔ se deve escrever ◊
is to be $1/en ↔ se deve $1/ir	is to be driven ↔ se deve conduzir
is to be $1/ed/en ↔ se deve $1/ar/er/ir	*(matches all of the above)*

Table 6.3: Six English-Portuguese translation patterns with different suffix restrictions equivalently rewritten as a single compact pattern. This illustrates how effectively the syntax for alternative suffixes reduces the number of patterns needed. ◊ The stem of "written" is "writ" and the matched suffix is "en", which means that the second "t" is not part of either of these. This is perfectly fine. Suffixes in variables are not required to match all characters after the stem.

To complement the previous pattern, we might also write the pattern "is to be $1/ed/en"↔"deve-se $1/ar/er/ir", which matches phrase pairs such as "is to be classified"↔"deve-se classificar".

6.2.6 Lists of Alternative Literals

Analogous to the compact representation of multiple prefixes and suffixes, we have a compact representation for multiple literals that may occupy the same position in a pattern. We list them separated by vertical line characters (|) and without spaces. For

example, the pattern "we are $1/ing you"↔"te|vos|lhe|lhes $1/mos", will match phrases such as "we are asking you"↔"vos pedimos" and "we are sending you"↔"lhe enviamos".

Pattern	Example Phrase Pair
$1/ly ↔ de forma $1	reliably ↔ de forma fiável
$1/ly ↔ de modo $1	generaly ↔ de modo geral
$1/ly ↔ de maneira $1	differently ↔ de maneira diferente
$1/ly ↔ de forma|maneira|modo $1	*(matches all of the above)*

Table 6.4: Three English-Portuguese translation patterns with different literals equivalently rewritten as a single compact pattern.

6.2.7 Negation Lists

Instead of specifying a list of required suffixes, prefixes or literal words in a pattern, as above, it is sometimes easier to specify a smaller list of *forbidden* suffixes, prefixes or literals, respectively.

For example, in the pattern "shall not $1!ly"↔"não $1/am/em/e/a/ará/arão/erá/erão/irá/irão", the variable $1 may be matched by any English word that does not end with the suffix "ly", which are typically adverbs, such as "likely". Therefore, this pattern matches phrase pairs such as "shall not deduct"↔"não deduzirão", "shall not hold"↔"não terá", "shall not exhibit"↔"não revele" and "shall not allow"↔"não permitem".

6.2.8 Context Restrictions

Sometimes it is useful to restrict the matching of a pattern to certain contexts. For example, consider the generic pattern "$1 $2"↔"$2 $1" which matches phrase pairs such as the ones highlighted in blue and orange in Figure 6.1:

... such as the Taric code ; ... ⇔ ... tal como o código Taric ; ...

... an international jury chaired by ... ⇔ ... um júri internacional presidido por ...

... known as « Herstatt risk » ... ⇔ ... conhecido como « risco Herstatt » ...

... entered into force on 1 January ... ⇔ ... entrou em vigor em 1 de Janeiro ...

... period from October 2008 to ... ⇔ ... período entre Outubro de 2008 e ...

Figure 6.1: Example matches of a too-general pattern, seen in context.

The last two phrase pairs, highlighted in orange, *are not* translation equivalents, despite matching the pattern above. In the penultimate phrase pair, the variable $1 matched the words "force"↔"vigor" and $2 matched "on"↔"em", and in the last phrase pair, $1 matched "from"↔"de" and $2 matched "October"↔"Outubro".

We can avoid these errors by adding contextual restrictions which will narrow down the contexts where this pattern is allowed to match. Looking closely at the immediate

contexts where the pattern was matched, we see that the correct matches, highlighted in blue, are immediately preceeded by equivalent words. For example, in the first line we see that the matched phrases are preceeded by "the"↔"o" and followed by ";"↔";". By contrast, incorrect matches, highlighted in orange, were preceeded and/or followed by non-equivalent words. For example, the match in the last line is preceeded by "period"↮"entre". If we turn this observation into a contextual restriction, then we will likely avoid most incorrect matches while still allowing most correct ones to be matched.

Contextual restrictions are added by surrounding our original pattern with *capture parentheses* and then adding literals or variables outside these parentheses. To distinguish capture parentheses from regular parentheses, we write capture parentheses as two opening/closing parentheses together "((...))". These *double parentheses* should be separated from the surrounding tokens with a space.

For example, we might replace the pattern "$1 $2"↔"$2 $1", which is too general, with the pattern "$1 (($2 $3)) $4"↔"$1 (($3 $2)) $4". If we look inside the capture parentheses, "... (($2 $3)) ..."↔"... (($3 $2)) ...", we see that this pattern is equivalent to the previous pattern, except for the variable numbers, i.e. both patterns express the same word reordering. Remember that, by convention, variables are numbered sequentially starting from 1 on the left-hand side. Thus, because we added variables outside the capturing parentheses, the original variables had to be renumbered.

Let us now analyse the newly added context restrictions of this pattern. The variables outside the capture parentheses, "$1 ((...)) $4"↔"$1 ((...)) $4", require that the phrases to be extracted are preceeded by a pair of equivalent phrases, matched by $1, and are also followed by a pair of equivalent phrases, matched by $4. The phrases matching $1 and $4 will not be part of the extracted phrases, since they lie outside the capture parentheses.

With this context-restricted pattern, all correct phrases, highlighted in blue, in Figure 6.1 are matched while incorrect ones, highlighted in red, are avoided.

6.3 State-of-the-Art

Translation patterns share similarities to both the *alignment templates* proposed by Och and Ney [85] and the *hierarchical phrases* employed by Chiang [19] in hierarchical phrase-based statistical machine translation (HPBSMT). In fact, translation patterns are rules that combine features from both alignment templates and hierarchical phrases, but also some features that are not present on either of these.

A fundamental difference between these approaches and the extraction method that will be discussed later in this chapter is that, while alignment templates and hierarchical phrases were both developed in the context of machine translation generation, here our goal is to recognize and extract phrase translations from phrase-aligned parallel texts. Thus, in this chapter we will be solving a *recognition* problem instead of a *generation* one.

6.3.1 Alignment templates

In the phrase-based approach to statistical machine translation (PBSMT) proposed by Koehn et al [56], phrase translations are extracted from a word-aligned parallel corpus, scored, and later used as translation units when the system translates new sentences. Observing that many phrases share a common word-level alignment, which is often determined by the syntactic categories of words, Och and Ney proposed the *alignment template* [85] approach to machine translation with the goal of generalizing extracted phrase pairs. An example alignment template, using a slightly adapted version of our pattern syntax, would be "$1[JJ] $2[CN]"↔"$2[CN] $1[JJ]", where "JJ" and "CN" are part-of-speech (PoS) tags, denoting an adjective and a common noun, respectively. This template would match phrase pairs such as "social policy"↔"política social" or "blue flower"↔"flor azul".

Note that, instead of PoS tags, Och and Ney employed word classes obtained with an unsupervised bilingual word clustering method that had been earlier proposed by Och [84]. Similarly to PoS tags, these unsupervised classes are expected to group together words that belong to the same syntactic and/or semantic category. Och and Ney give the following example: if an alignment template is extracted from a phrase that contains a town name, then it is expected that the template generalizes that phrase translation to other town names. However, because unsupervised classes do not have human-friendly descriptors (they are identified by automatically assigned numbers) we cannot present a human-intelligible alignment template using those classes.

Alignment templates are automatically mass-extracted from a word-aligned parallel corpus, using a slightly adapted version of the algorithm used to extract phrase pairs, which we already discussed in the context of phrase alignment in Subsection 4.2.1. The adaptation of the algorithm consists of two minor modifications: (1) word alignments within the phrases are used to determine the numbering of the variables in the extracted templates and (2) instead of words, the algorithm extracts phrases of word classes. Because templates are extracted with the same algorithm employed for extracting phrase pairs, they are subject to the same problems known to plague phrase tables of PBSMT systems: high redundancy and lack of proper segmentation. Both problems are closely related. High redundancy means that a large number of templates is extracted, but the same results could be obtained, more efficiently, with a smaller set of templates. Lack of proper segmentation of phrases (and templates) results from the design of the extraction algorithm, which attempts to extract translations for every possible sequence of tokens in the corpus. Extracted alignment templates are probabilistically scored and filtered, and then used as part of a statistical translation model [85]. By contrast, our patterns are carefully written by hand and thus are in much smaller number, but as we will see, extremely precise and productive.

If we assume that our morphological restrictions (prefixes and suffixes) are more or less equivalent to PoS tags in alignment templates, then we may say that alignment

templates are a subset of what can be expressed by translation patterns, because:

1. Variables of alignment templates can only be instantiated by words, while in our patterns unrestricted variables can be instantiated by phrases;
2. Alignment templates do not have literal words;
3. Alignment templates do not have unrestricted variables;
4. Alignment templates do not have context restrictions.

As we will see further ahead, our extractor implementation can be easily extended to allow PoS-based restrictions in addition to the morphological restrictions. This is an open possibility for future experimentation. However, there are advantages to the use of morphological restrictions. First, any person can read and write patterns with prefix and suffix restrictions with no need to learn the PoS tagsets for the languages involved. Second, the availability and accuracy of automatic PoS taggers varies from language to language. While there are good taggers available for English and some European languages, there are many languages for which no tagger is available. Thus, the sub-word morphology-based solution is potentially applicable to a larger number of languages.

6.3.2 Hierarchical Phrases

Hierarchical phrases, proposed by Chiang [19, 20], generalize phrase translations in a different way than alignment templates. Instead of phrases of word classes, hierarchical phrases are phrases of literals intermixed with variables, like our patterns. While in the case of alignment templates the generalized phrases are abstract sequences of restricted variables that can only be instantiated by words of specific classes, in the case of hierarchical phrases, the phrases contain concrete words (literals) intermixed with variables, but these variables are much more generic as they are allowed to match any phrase translation, like unrestricted variables in our patterns. Therefore, alignment templates are more *rigid* than hierarchical phrases because they require exact matches of sequences of word classes. For example, these two phrase translations do not match the same alignment template simply because they have different number of words[1]:

visiting New York ↔ de visita a Nova Yorque
visiting Lisbon ↔ de visita a Lisboa

By contrast, the hierarchical phrase "visiting $1"↔"de visita a $1" would match both phrases given above. This added flexibility of variables in hierarchical phrases (being allowed to match phrases of any length) has the downside of increased ambiguity when variables appear at the beginning or at the end of phrases.

For example, for the sentence "She was visiting Lisboa when Portugal won Euro 2016", there are several possible instantiations for the variable in the hierarchical phrase above:

[1] It is debatable what *constitutes* a word. Is *database* a word or two? Not so long ago, *database* was spelled *data base*. For the ongoing discussion let us follow the usual assumption that whitespace is a word delimiter, aknowledging that punctuation is commonly glued to words and some whitespace-separated tokens belong in fact to the same word.

Feature/Property	Alignment Templates	Hierarchical Phrases	Translation Patterns
Automatically extracted	yes	yes	no*
PoS-based restrictions	yes	no	no**
Morphology-based restrictions	no	no	yes
Variables match phrases	no	yes	yes
Context restrictions	no	no	yes
Literals	no	yes	yes

Table 6.5: Design choices of alignment templates, hierarchical phrases and translation patterns. (* only simple patterns for now, to be further developed in future work; ** not planned, but would be trivial to implement)

Lisboa
Lisboa when
Lisboa when Portugal
Lisboa when Portugal won
Lisboa when Portugal won Euro
Lisboa when Portugal won Euro 2016

As a consequence of this matching ambiguity, hierarchical phrases for translation purposes are generally limited in several ways:

- Only a limited number of variables is allowed (typically at most 2 variables).
- Variables are not allowed to be together in the source language (must have at least one word between them).
- Variables are not allowed at the beginning or end of hierarchical phrases in the source language.

As is already clear, like alignment templates, hierarchical phrases are also a proper subset of our pattern language. Table 6.5 presents a comparison summary of the design choices of the three approaches.

6.3.3 Pattern-Based Extraction From Monotonic Alignments

As mentioned earlier in the chapter dedicated to phrase alignment (Chapter 4), the work presented on this thesis, although new, follows from earlier research carried out within the research group led by my supervisor, Professor Gabriel Pereira Lopes. Of particular relevance to this chapter is the phrase translation extraction method proposed by Aires et al [5], which employs hard-coded patterns designed to exploit the monotonic alignments produced by the method described in my MSc thesis [40], which in turn follows from earlier research within our group [48, 94].

That extraction method employed three specific patterns that we will illustrate using the monotonic alignment shown in Figure 6.2.

#	English	Known?	Portuguese
1	Eurojust's		A Eurojust tem por
2	mission	*	missão
3	shall be to support		apoiar
4	and	*	e
5	strengthen		reforçar
6	coordination and cooperation	*	a coordenação e a cooperação
7	between	*	entre
8	national investigating and prosecuting		
9	authorities	*	as autoridades
10			nacionais competentes para a investigação e o exercício de a acção penal
11	in relation to	*	em matéria de
12			criminalidade
13	serious	*	grave
14	crime		
15	affecting	*	que afecte
...	...		...

Figure 6.2: Pattern-based extraction applied to monotonic phrase alignments. Adapted from [5].

The first pattern exploits alignments where a phrase on the left-hand side was aligned with an empty segment on the right-hand side, as in segment 8 of the figure, followed by a pair of known phrase translations, as in segment 9, followed by a segment that contains an empty phrase on the left-hand side aligned with a phrase on the right-hand side, as in segment 10. Therefore, in the example of Figure 6.2 this first pattern would match the phrases "national investigating and prosecuting authorities"↔"as autoridades nacionais competentes para a investigação e o exercício de a acção penal", which are long but, nevertheless, correct translations of each other. This pattern is illustrated by Figure 6.3(a).

The second pattern exploits situations symmetric to the previous one, as illustrated by Figure 6.3(b). This pattern would match the phrases "serious crime"↔"criminalidade grave".

The third pattern matches non-empty segments that sit between consecutive anchors as illustrated by Figure 6.3(c). Continuing with the example of Figure 6.2, this pattern matches the phrases "strengthen"↔"reforçar", "Eurojust's"↔"A Eurojust tem por" and "shall be to support"↔"apoiar".

Obviously, not all phrase pairs matched by these three patterns are correct translations. To filter out incorrect translations, Aires et al [5] employ scores based on occurrence and co-occurrence frequencies of the phrases.

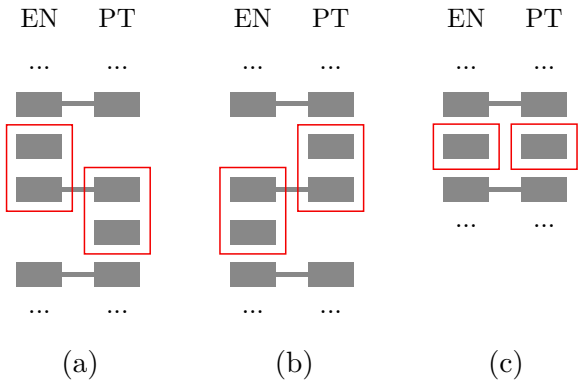


Figure 6.3: First extraction pattern applied to monotonic phrase alignments.

The three patterns described above may be expressed in the language of translation patterns earlier described.

The first pattern is expressed as: ""$1 (($* $2)) $3"↔"$1 (($2 $*)) $3"" Dissecting this pattern, the variables $1 and $3 represent the anchors that must exist immediately before and after the extracted phrases, in order to delimit the unknown segments that are matched by the unlinked variables $* in both sides of the pattern. The variable $2 represents the anchor segment that sits between the two empty-aligned phrases.

Revisiting the example of Figure 6.2, variables $1, $2 and $3 would match the anchor segments 7, 9 and 11; the unlinked variable $* would match segment 8 on the left-hand side and segment 10 on the right-hand side. As a result, the segments 8, 9 and 10 would be matched within the capture parentheses, while the segments 7 and 11 would be matched by the pattern, but not included in the extracted phrase pair.

The second pattern is symmetric to the first, with respect to the placement of the unlinked variable and variable $2:

""$1 (($2 $*)) $3"↔"$1 (($* $2)) $3"".

This pattern would match segments 11 and 15 to variables $1 and $3, respectively, and segment 13 to variable $2.

Finally, the third pattern is expressed as:

""$1 (($*)) $2"↔"$1 (($*)) $2"".

This pattern will match phrase pairs that occur within two monotonically-aligned phrase pairs, as is the case of segments 1, 3 and 5, of which only the last one is a correct translation.

The possibility of creating new patterns, trying and adjusting them as necessary to improve their accuracy, all without having to write a single line of code is one of the main benefits of the new pattern-based extraction method proposed in this chapter. Another advantage is that, being based on the non-monotonic phrase alignments described in the previous chapter, this method is also much more precise, as we will see later.

Czech	Portuguese	English
prac·ovat	trabalh·ar	work·
prac·uji	trabalh·o	work·
prac·uješ	trabalh·as	work·
prac·uje	trabalh·a	work·s
prac·ujeme	trabalh·amos	work·
prac·ujete	trabalh·ais	work·
prac·ujou	trabalh·am	work·
tanc·ovat	danç·ar	dance·
tanc·uji	danç·o	dance·
tanc·uješ	danç·as	dance·
tanc·uje	danç·a	dance·s
tanc·ujeme	danç·amos	dance·
tanc·ujete	danç·ais	dance·
tanc·ujou	danç·am	dance·

Table 6.6: Example word translations sharing equivalent stems and suffixes.

6.3.4 Translation Generation

Consider the word translations shown in table 6.6. Translations in the top group have the same suffixes as translations in the bottom group. Within each group, all translations share the same stem pairs. Thus, if we had these examples in our lexicon then we could extract stem and suffix equivalences that later would help us match other word translations. For example, from the Czech and Portuguese translations we would extract "prac⋆"↔"trabalh⋆", "tanc⋆"↔"danç⋆", "⋆ovat"↔"⋆ar", "⋆uji"↔"⋆o", "⋆uješ"↔"⋆as", etc.

While in this mini-lexicon we have more distinct suffix pairs than stem pairs, the fact is that the number of distinct suffix pairs in any language pair is much lower than the number of stem pairs.

While the problem addressed in this chapter is an extraction problem, the sub-word equivalence in the examples above can also be exploited for translation generation. In this perspective, a word may be translated by replacing its stem and suffix by equivalent counterparts in the target language. This brings to mind the decoding approach to machine translation envisaged by Warren Weaver in this chapter's epigraph.

Kavitha Mahesh embraced the translation generation problem and developed methods for extracting bilingual stem and suffix pairs from a bilingual word lexicon [73], clustering bilingual suffix pairs based on the stems they attach to [75], and classifying words with respect to the learned suffix clusters. Then, by concatening bilingual stem pairs with suffix pairs of the correct clusters, Kavitha Mahesh was able to generate translations for many word forms that were missing from the lexicon [70, 72]. This line of work is fully described in Kavitha Mahesh's PhD thesis [69], recently defended.

6.4 Proposed Method for Matching Translation Patterns

The matching of translation patterns is implemented as *recursive alignment candidate generator*[2] within the phrase alignment method proposed in Chapter 4. We call this generator *pat_match*, which stands for *pattern matcher*.

As explained in Subsection 4.3.7, inexact matching of the lexicon based on word stems enables matching TEs with a wider range of word forms than those in the lexicon. Unsurprisingly, this stem-based matching frequently generates alignment candidates that are not exactly correct.

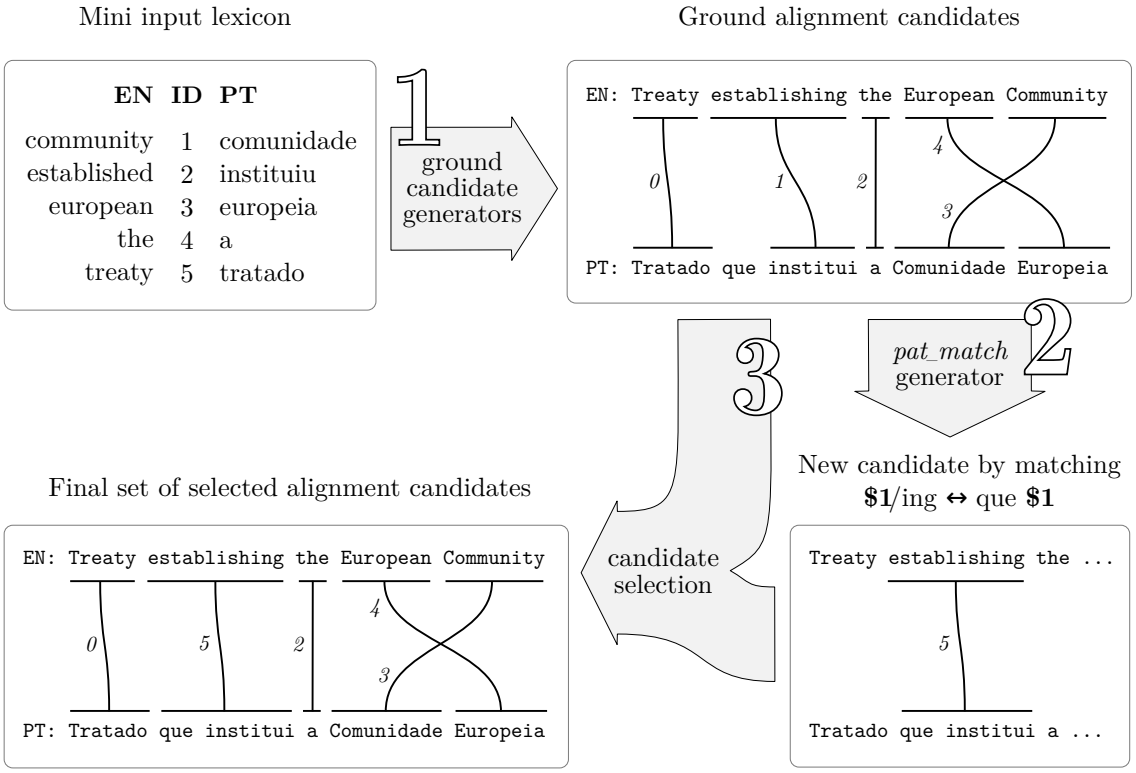


Figure 6.4: Context of *pat_match*, showing its input and output and how it afects the final alignment. Steps are numbered according to the order they are executed.

For example, consider the alignment candidates represented at the top-right corner of Figure 6.4. These candidates were generated by ground generators using the mini lexicon shown at the top-left corner of the figure. Candidate 1, in particular, was generated by the *stem_lookup* generator by matching the stems "establish∗"↔"instit∗" of the TE with ID 2. However, this candidate is not exactly correct. The correct alignment would be "establishing"↔"que institui", which is not in the given input lexicon.

This specific type of alignment error can be fixed with the help of the translation pattern "$1/ing"↔"que $1". In these example sentences this pattern would match the phrase pair "establishing"↔"que institui", as intended, but in other sentences it would

[2]Ground and recursive generators were explained in Subsection 4.3.3.

fix similar errors for other verbs. Candidate 1, although not correct, is essential because it gives us the stem equivalence that we need to match variable $1 in the pattern.

Thus, after we add this pattern to our lexicon, the *pat_match* candidate generator which we are about to describe, would use candidate 1 to match variable $1 in the pattern and thus generate candidate 5, which aligns "establishing"↔"que institui". Since the newly generated candidate 5 strictly subsumes candidate 1, it will have higher coverage and the selection step of the aligner would select it instead of candidate 1, as shown in the final alignment at the bottom left corner of the figure.

A single translation pattern may affect alignments for many sentences in a corpus or only a few, depending on how specific it is. Consequently, a single pattern may have a significant impact in the overall quality of alignments.

Now that we have seen the input and output of *pat_match* and how it is integrated into the phrase alignment method proposed earlier, let us move on to the implementation of *pat_match*.

6.4.1 Overall Algorithm

The translation patterns language is a high-level language that was designed to be human friendly. To perform the matching, we will compile this high-level language into a lower-level pattern language for which there are many efficient implementations available. Specifically, we will use the regular expression (regex) language[3], which is briefly described in Appendix B.

The overall algorithm for matching translation patterns has three steps. The first one is to compile each translation pattern into a regex. This step is executed only once for each new translation pattern added to the lexicon. Once compiled, the regex is stored and reused in the future. The second step is to generate a textual representation of alignment candidates embedded within the input sentences. The third step is to employ the regexes generated in the first step to find matches within the representation generated in the second step. Each match found will give rise to a new alignment candidate.

The key factor to understanding this solution is the textual representation of alignment candidates embedded within parallel sentences. Therefore, we will look first into this representation and then we will be in a better position to understand the compilation of translation patterns into regexes that will be matched against this embeded representation.

6.4.2 Embedding Alignment Candidates within Parallel Sentences

The syntax for embedding alignment candidates within parallel sentences has two rules:

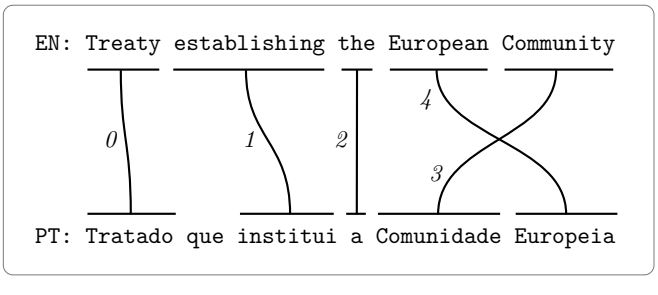
1. the two input sentences are written one after the other and separated with the special tag `<=>`;

[3]The current implementation employs the Python regex library `https://pypi.python.org/pypi/regex/`.

2. text segments of alignment candidates are wrapped within pairs of tags of the form `<id>...</id>`, where *id* is the candidate identifier.

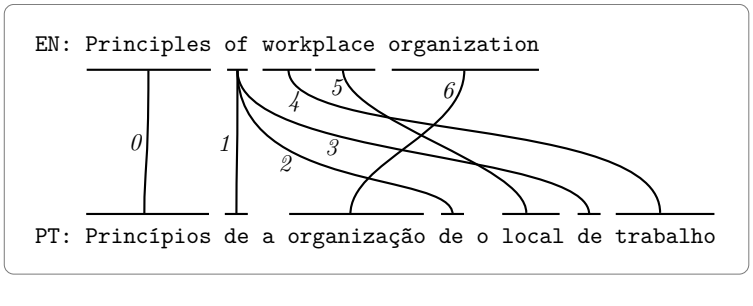
Figure 6.5 shows the resulting embedded representation of the alignment candidates from the previous example and Figure 6.6 shows the embedded representation for a slightly more contorted and interesting pair of sentences, which we will use in forthcoming examples.

Remember that the input for *pat_match* is a set of candidates that have not yet been subject to the selection step. Therefore, many of these candidates are incorrect alignments. For example, in Figure 6.6 there are multiple candidates for preposition "of" but only candidate 1 is correct. Nevertheless, the embedded representation of alignment candidates will have all candidates represented.




```
<0>Treaty</0> <1>establishing</1> <2>the</2> <3>European</3> <4>Community</4>
                                  <=>
<0>Tratado</0> que <1>institui</1> <2>a</2> <4>Comunidade</4> <3>Europeia</3>
```

Figure 6.5: Parallel sentences with embedded alignment candidates.




```
<0>Principles</0> <1><2><3>of</3></2></1> <4><5>workplace</4></5> <6>organization</6>
                                          <=>
<0>Princípios</0> <1>de</1> a <6>organização</6> <2>de</2> o <5>local</5> <3>de</3> <4>trabalho</4>
```

Figure 6.6: Parallel sentences with embedded alignment candidates for a compound word.

The embedded candidate representations are shown in multiple lines because they would not fit in a single line. However it is important to bear in mind that these lines are in fact a single string. If they were separate strings we would not be able to employ

back-references to enforce matching of tags with the same number in both sentences, as explained further ahead.

Unlike HTML tags, candidate tags are not always arranged in a hierarchical manner. A case in point are the tags for candidates 4 and 5 around the word "workplace" in Figure 6.6. These candidates were generated by *stem_lookup* which matched the stems "work" and "place". Although each of these candidates matches only a part of the word "workplace", their tags are written around the whole word and it is very important that the opening tags and closing tags are written in the same order as the corresponding stems appear within the word. The non-hierarchical arrangement of these tags indicates that the matched segments do not overlap each other, which is true because the stems appear one after the other in the word.

By contrast, a hierarchical arrangement of tags indicates that the candidate represented by the outer tags subsumes the candidate represented by the inner tags. Note that if two or more candidates have exactly the same segment on one of the languages, such as candidates 1, 2 and 3 in Figure 6.6, then each one subsumes the others (but not strictly) and vice versa. Thus their tags must be arranged hierarchically, but the order is irrelevant.

6.4.3 Compilation of Translation Patterns Into Regexes

The name *regular expressions* derives from the fact that regular expressions define *regular languages* which, in formal language theory, are the languages that can be described by *context-free* grammars. However, today's regex compilers have many features that far exceed the expressive power of regular languages. Of particular interest to us is the *back-reference* feature, which is fundamental to the implementation of our translation pattern matcher. Any regex that includes back-references does not describe a context-free language. Instead, it describes a *context-dependent* language, which is in fact the case of our language.

For the readers that are not familiar with regexes or wish to refresh their memory, Appendix B provides a quick introduction, focusing on the features that are relevant to the presentation that follows.

Translation patterns have been presented with each language side enclosed in quotes and using double-arrows (↔) to connect both sides. But in real use, they are written without quotes and the string `␣<=>␣` is used as a separator of both language sides, where `␣` is the whitespace character.

To compile a translation pattern into a regex we first *tokenize* it into a sequence of tokens. A token is our unit of compilation. Each token is translated into a short regex and the final regex is obtained by concatenating all regexes of individual tokens, as depicted in Figure 6.7.

The separator is itself considered a token and it is compiled into the regex `.* <=> .*`, where the dot-star (`.*`) matches any sequence of zero or more characters and the separator

`␣<=>␣` matches itself, since none of its characters has special meaning in regexes. The compilation of each token type is explained in the next subsubsections.

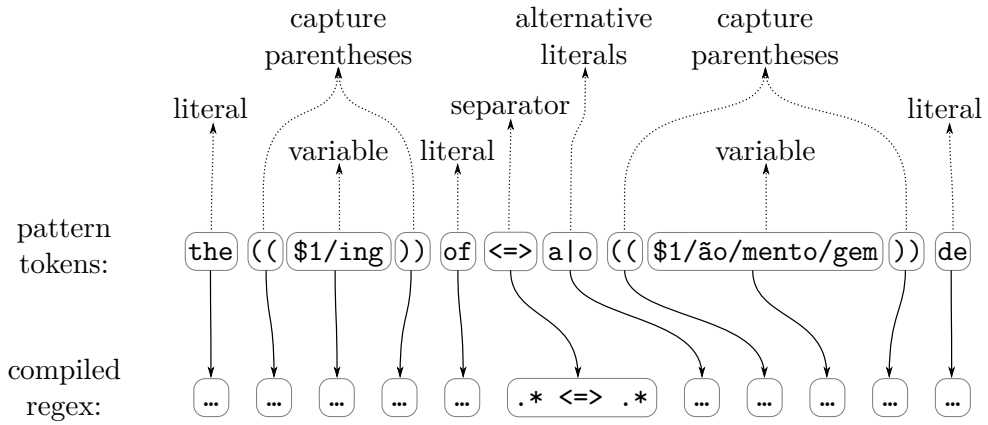


Figure 6.7: Compilation of a pattern by compiling each token individually.

Capture Parentheses

Capture parentheses define what part of the matched phrases is to be extracted as a TE. For example the pattern "the (($1/ing)) of"↔"o|a (($1/ão/mento/gem)) de" will extract as TE only the words that match the variables, although the whole pattern must be matched.

This pattern will match the phrases "the steering of"↔"a orientação de", "the emerging of"↔"o surgimento de" and "the approaching of"↔"a abordagem de", but will extract the word pairs "steering"↔"orientação", "emerging"↔"surgimento" and "approaching"↔"abordagem", respectively.

If a pattern does not have explicit capture parentheses, then they are implicitly placed around all tokens of each language side. For example, the pattern "$1$2"↔"$2 de $1" is equivalent to "(($1$2))"↔"(($2 de $1))".

Capture parentheses in a pattern are translated as named capture groups in a regex, as depicted in Figure 6.8. The names of the capture groups for the left- and right-hand sides of the pattern are *p1* and *p2*, which stand for phrase 1 and 2, respectively.

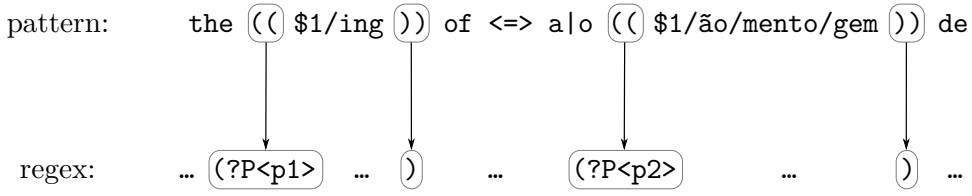


Figure 6.8: Compilation of capturing parentheses into regex capture groups.

Literals

Literals in patterns are strings that match themselves in text. The syntax for specifying lists of alternative literals is translated into regexes using the alternation syntax enclosed within a non-capturing group.

In Figure 6.9 we see how the literals "the", "of" and "de" are translated as themselves in the regex, while the list of alternative literals "a|o" is translated as an alternation within a non-capturing group.

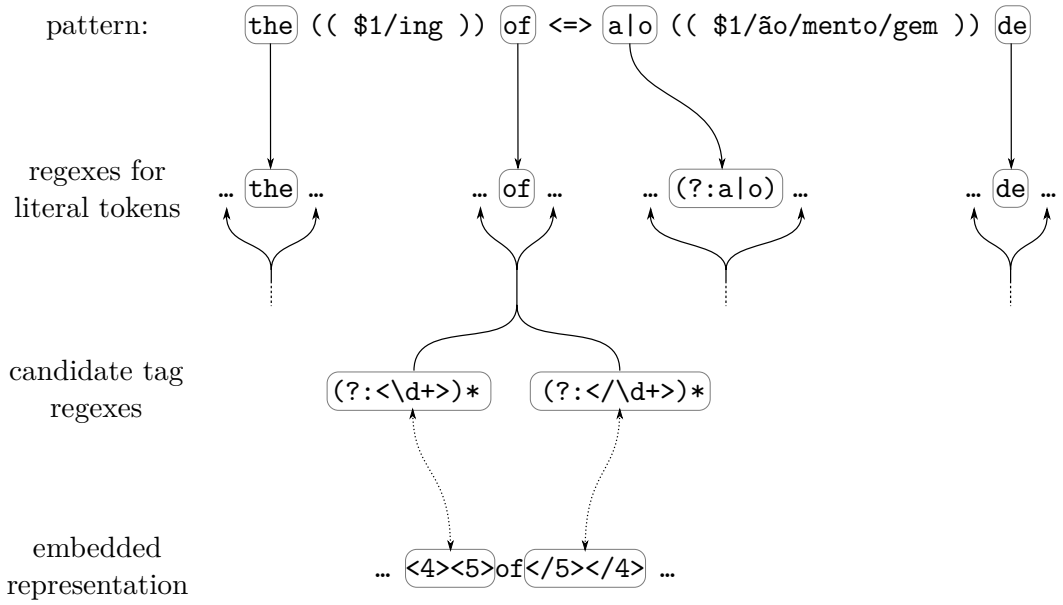


Figure 6.9: Compilation of literals.

A literal token is its own regex, but care is needed for characters that have a special meaning in the regex language, such as the asterisk, parentheses, dot, etc. These characters must be *escaped* by prepending them with a backslash character (\). Escaping removes their special meaning and makes them match themselves. For example, if a literal token contains a plus sign `+`, which as explained in Appendix B, is a regex quantifier, then we would translate it as `\+`, which matches a plus sign.

The regex of each literal token must be wrapped within regexes `(?:<d+>)*` and `(?:</d+>)*`. The construct `(?:...)` in these regexes defines a non-capturing group and the asterisc next to it specifies that the regex within the group matches zero or more times. Thus, these regexes match zero or more candidate tags around each literal token in the embedded candidates representation, as depicted in the lower half of Figure 6.9.

Morphologically Restricted Variables

Morphologically restricted variables are variables that have prefix and/or suffix restrictions, such as `$1/ing`.

A variable on the left-hand side of the pattern defines a capture group named *c#*, where # is the number of the variable. This capture group will match the ID of an alignment candidate. The corresponding variable on the right-hand side, will contain back-references to this named capture group, ensuring that the same candidate is matched on both language sides.

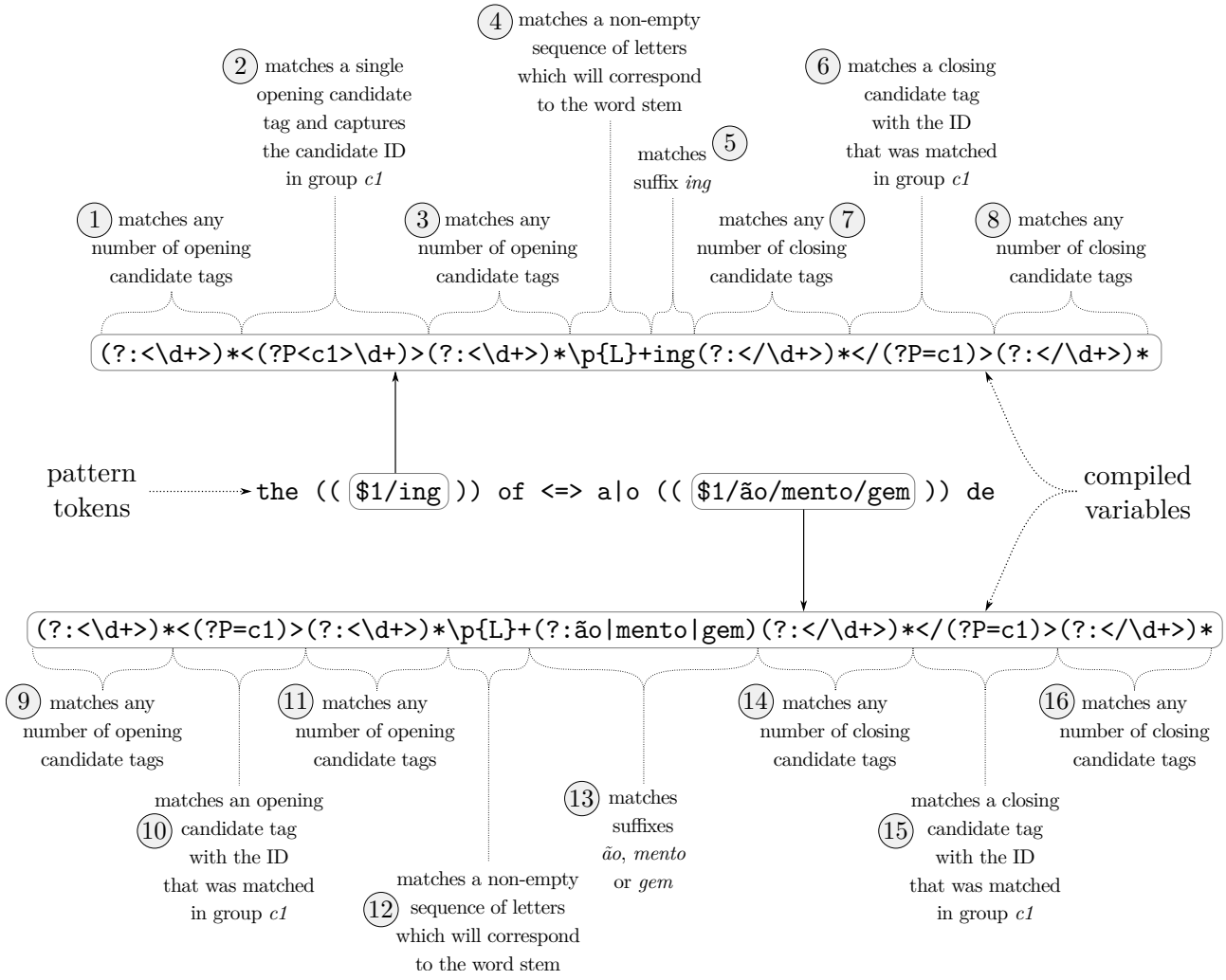


Figure 6.10: Compilation of morphologically restricted variables.

Figure 6.10 shows a example pattern with suffix-restricted variables. The translation pattern is placed at the center and the regexes for the left- and right-hand side variables, are shown at the top and bottom of the figure, respectively. Each of these regexes is subdivided in smaller blocks, each one described in the figure and numbered in the same order that the regex engine will attempt to match them.

From a high-level perspective, each of these two regexes matches a single word wrapped within candidate tags. The regex of the whole translation pattern will only match if the same candidate ID is matched by the capturing group *c1*, which is defined in block 2 and back-referenced in blocks 6, 10 and 15. In blocks 4 and 12 we see the

regexes responsible for matching the stems of these variables. Regex `\p{L}` matches a letter character, and the `+` indicates that we want to match a non-empty sequence of letters. In blocks 5 and 13 we see how suffix restrictions are compiled into regexes. When there is only one suffix, we write it unmodified next to the regex of the stem, as in block 5. When there are multiple suffixes, we write them in an alternation within a non-capturing group next to the regex of the stem, as in block 13.

Prefix-restricted variables will result in similar regexes, but the prefixes are written before the stem regex instead of after.

When a variable is restricted with a list of forbidden prefixes or suffixes, we will use a negative lookahead or lookbehind, respectively. For example, the prefix and stem of variable "!un\$1" would be compiled as `(?!un)\p{L}+`, and the stem and suffix of variable "$1!ly" would be compiled as `\p{L}+(?<!ly)`.

Unrestricted Variables

Unrestricted variables are variables without prefix or suffix restrictions, such as for example the variables in the pattern "$1 $2"↔"$2 de $1". These variables are allowed to match words or phrases.

The compilation of unrestricted variables is similar to the compilation of morphologically-restricted variables except that the former do not contain prefix of suffix and instead of the regex `\p{L}+` for matching the stem of a word, we employ `.+` which matches a non-empty sequence of any characters, including the whitespace characters.

Thus, reusing the regex blocks presented in Figure 6.10, a variable $1 on the left-hand side of a pattern would be compiled into a regex composed of blocks 1 to 8, but replacing `\p{L}+` by `.+` (a dot) in block 4, and deleting block 5. Analogously, a variable $1 on the right-hand side of a pattern would be compiled into blocks 9 to 16, but replacing `\p{L}+` by `.+` in block 12, and deleting block 13.

Glued Variables

Glued variables are used to represent stems within compound words. For example, the variables on the left-hand side of pattern "$1$2"↔"$2 de $1" are glued together. This pattern would match TEs such as "sunglasses"↔"óculos de sol", "teacup"↔"chávena de chá" and "snowball"↔"bola de neve".

Remember, from the Subsection 6.4.2, that tags for multiple stems matched within a single word are written before and after the word. For example, we saw in Figure 6.6 that the representation for candidates 4 and 5, which resulted from matching the stems "work" and "place" within the word "workplace" would be written as `<4><5>workplace</4></5>`.

The regex of a series of glued variables is similar to a regex of a restricted variable, except that instead of matching a single pair of candidate tags the regex will match one pair of tags for each variable, and the order of the tags must be the same as the order

Figure 6.11: Compilation of glued variables.

of the variables. For example, in Figure 6.11 we see the compiled regex for variables "$1$2" if they appeared on the left-hand-side of a pattern. This regex is presented in two lines because it does not fit in a single line. If the same variables appeared in the right-hand side of a pattern, the only difference would be that blocks 2 and 4 would contain back-references to the named capture groups *c1* and *c2*, respectively.

This concludes the description of compilation of translation patterns into regexes, and the description of the pattern matching method.

6.5 Evaluating Pattern-Based Extraction

This section presents the results of a relatively large-scale experiment, covering ten language pairs, including challenging languages such as Czech (morphologically rich), German (highly agglutinative and morphologically rich) and Mandarin (no explicit word segmentation).

L1-L2	Language 1-Language 2	# Sentences	# L1 Tokens	# L2 Tokens
CS-PT	Czech-Portuguese	3,965,841	63,367,038	81,653,296
DE-EN	German-English	4,176,882	67,875,079	73,012,986
DE-ES	German-Spanish	3,825,587	61,282,915	76,053,965
DE-PT	German-Portuguese	3,898,384	62,935,069	80,226,781
EN-ES	English-Spanish	4,134,447	71,492,735	82,324,167
EN-FR	English-French	4,649,885	84,398,620	104,757,870
EN-PT	English-Portuguese	4,563,280	79,904,700	94,274,091
EN-ZH	English-Mandarin	72,339	3,043,756	4,319,791
ES-PT	Spanish-Portuguese	3,926,732	77,904,255	80,062,941
FR-PT	French-Portuguese	4,143,850	92,028,903	87,468,633

Table 6.7: Corpora used in the pattern-based extraction experiments.

6.5.1 Corpora and Languages

The ten language pairs included in these experiments are listed in Table 6.7 in alphabetic order of the respective language codes (the leftmost column).

Remind that all methods proposed in this thesis are agnostic with respect to translation direction, i.e. they work the same independently of which language we choose to put at the left- or right-hand side, provided that all data files respect the same order. By convention, we choose the left- and right-hand languages based on the lexicographic ordering of the respective language codes. Thus, we choose EN-PT instead of PT-EN.

For the nine pairs of European languages we used the DGT-TM parallel corpus[104] together with the OPUS EMEA corpus [107]. These two corpora are composed of texts from European institutions. The first contains mostly legislation and policy domains, while the second is of the more specialized domain of medicines.

For the English-Chinese (Mandarin) language pair we used the UN corpus [91], which is composed of parliamentary documents of the United Nations that are in the public domain.

These corpora have been extensively used in research in the field of statistical machine translation, and are described in full detail in the respective publications [91, 104, 107], which we will not reproduce here. For our purposes, the most important property of these corpora is their size, which is summarized in Table 6.7, both in terms of parallel sentences and tokens for each language. Note that for Mandarin, each character was counted as a token since in this language words are not separated by whitespace[4]

Not only the corpora used for the European languages is much larger than the corpus used for EN-ZH, but also more human resources were available for each of the European languages than for EN-ZH.

[4]As a pre-processing step, we introduce whitespace characters between Chinese characters, so that each character is seen as a token by the phrase alignment method.

6.5.2 Experimental Procedure

In the previous section we described a method for matching translation patterns. While this is the most important part in extracting new TEs based on translation patterns, it is not a full extraction method on itself.

Like in Chapter 5, to extract new TEs we will combine our proposed matching method with a filter based on a co-occurrence similarity measure, specifically the Dice score. As mentioned earlier, Dice is one of the most accurate measures for TE extraction according to António Ribeiro [94].

The whole extraction and evaluation procedure is decomposed in the following steps, which was repeated as frequently as necessary to ensure that linguists always had new TEs to be validated during the time span of their contract:

1. Match translation patterns in each sentence pair of the parallel corpus with the method described in the previou section. Note that as a side-effect of this step, the corpus is re-aligned at phrase level.
2. Collect and count phrase pairs aligned by candidates that were generated by translation pattern matches and selected as final alignment links, over the entire corpus.
3. For each collected phrase pair, compute its Dice score based on the total number of times that it was aligned and the number of occurrences of each phrase in the corpus.
4. Filter out phrase pairs with Dice score lower than specified thresholds.
5. Manual validation by linguists of phrase pairs that passed the filter.

From initial experimentation with the EN-PT language pair, we found that the precision of extracted TEs was well above 90% even when we set the threshold of minimum co-occurrence similarity (Dice) to zero. However, we found that this filter is still useful when used with a low threshold value, such as 0.1, to filter out rare incorrect phrase pairs that the selection step of the aligner failed to reject[5]. Therefore, the filter threshold was set to 0.1 for all language pairs.

Since this method is able to extract word and multi-word TEs, and we do not know how many multi-word TEs exist in our parallel corpora, we cannot compute a meaningful recall value. Therefore, we will evaluate the pattern-based extraction primarily based on precision.

6.5.3 Overall Results

In Table 6.8 we have the total number of patterns employed for each language pair, the number of accepted and rejected TEs that were extracted with the pattern-based method, and the precision. The column named *Initial # TEs* contains the size of the initial lexicon which had been extracted by other methods before the rounds of pattern-based extraction took place.

[5] These situations occur mostly on sentences that are not parallel.

L1-L2	Initial # TEs	# Patterns	# Accepted	# Rejected	Precision
EN-PT	432,804	10,032	437,217	9,203	97.94%
DE-PT	81,139	1,569	170,504	13,877	92.47%
EN-FR	65,290	504	150,038	1,916	98.74%
EN-ES	113,231	3,442	144,157	9,616	93.75%
FR-PT	54,745	749	47,242	1,071	97.78%
ES-PT	75,156	55	31,906	224	99.30%
DE-ES	24,880	190	24,569	867	96.59%
DE-EN	103,091	170	24,094	3,119	88.54%
CS-PT	9,349	384	2,397	1	99.96%
EN-ZH	5,181	21	1,291	137	90.41%
Totals:		17,142	1,033,724	40,033	96.27%

Table 6.8: Results of the pattern-based extractor for ten language pairs, sorted by the total number of accepted lexicon entries.

The average precison across all languages, computed from the total number of accepted and rejected TEs, shown in the last line of the table, is 96.27%.

All language pairs except DE-EN have a precision above 90%. Taking into consideration the fact that when a second validator was added to the DE-EN language pair, near the end of the evaluation period, we observed a higher disagreement between the two validators in comparison to other language pairs that also had two validators, we suspect that a higher number TEs may have been incorrectly validated in this language pair. We do not discard the possibility of this lower precision be caused by difficulties intrinsic to the German language, but we do not observe the same low precision on DE-PT or DE-ES.

In the case of EN-ZH, the evaluation took much longer than for other language pairs, partially because the validator was not a Mandarin native speaker and often needed to consult a Mandarin dictionary (paper version), which takes a relatively long time. As a consequence, the number of validated TEs is much lower for EN-ZH than for all other language pairs. The precision obtained was still above our expectations, because the size of the initial lexicon was extremely small, with only 5181 TEs, and because this language pair is often regarded as being harder to align than European languages.

6.5.4 "Precision" of the Precision Figures

A precision figure, alone, might be misleading, as we can always choose to extract less quantity but with higher precision, for example by extracting only phrase pairs with the highest co-occurrence similarity. However, as shown in Table 6.8, not only we extracted and evaluated a very large number of TEs for some language pairs, but also, many extracted TEs occur only *once* or *twice* in the corpus.

To put the number of extracted TEs and the reported precisions into perspective, we performed an extraction from the same corpora using Anymalign [60], which is a state-of-the-art statistical aligner/extractor.

L1-L2	# Accepted	# Rejected	Precision
EN-PT	3284	1412	69.93%
FR-PT	1366	453	75.10%
EN-FR	1275	768	62.41%
DE-ES	1263	682	64.94%
DE-EN	1004	885	53.15%
EN-ES	762	461	62.31%
DE-PT	616	433	58.72%
ES-PT	557	148	79.01%
EN-ZH	187	307	37.85%
Totals:	10314	5549	65.02%

Table 6.9: Results of a state-of-the-art statistical extractor (Anymalign) for nine language pairs, sorted by the number of total accepted lexicon entries.

Anymalign extracted several million phrase pairs for each language pair, but given the overall lower precision of this method, we only validated a small subset of the extracted pairs, for comparison purposes. According to its authors, the output of Anymalign is sorted in a way that causes most errors to appear at the end of the file. Thus, to obtain a subset with the highest precision possible, we selected from Anymalign's output the top 5,000 pairs that were not already in our lexicon at the time.

Then, linguists were asked to spend two to three days validating TEs extracted by Anymalign until at least 1,000 TEs had been validated or the allocated time ran out. Table 6.9 shows the number of accepted and rejected TEs and the respective precisions. For all language pairs[6], the precision of extraction with Anymalign is much lower than the precision of extraction with translation patterns, even though we considered only the top 5,000 pairs of Anymalign's output. In principle, if we had considered the top 50,000 pairs instead of the top 5,000, the precision would be lower, because of the increasing noise as we move towards the end of Anymalign's output.

Extracting new TEs from a parallel corpus becomes increasingly harder as the lexicon size grows, because most of the easier-to-extract TEs are already in the lexicon and only the harder remain to be extracted. However, after the evaluation of Anymalign, we continued extracting with the pattern-based method and validating TEs for several months, and thus the pattern-based method was not in an advantageous position in this regard.

In defense of Anymalign, it is a fully automatic extractor while the pattern-based method requires someone to write patterns in the first place. Thus, since the two extractors have completely different natures they are not really comparable. But the goal of presenting this Anymalign evaluation is not to claim that one is better than the other. Instead, the goal is to demonstrate how hard it is to achieve high precision while extracting as many TEs as were extracted by the pattern-based method.

[6]When this experiment was carried out, CS-PT was not one of the language pairs being actively worked on, and thus it was not included.

6.5.5 Precision vs Frequency of Translation Equivalents

If we compare a the phrase pairs at the top of the output file of Anymalign [60], with those at the bottom of the file, we observe that:

1. most phrase pairs at the top are correct
2. most phrase pairs at the bottom are incorrect
3. phrase pairs at the top are frequent (more than 100 occurrences)
4. phrase pairs at the bottom of the file are rare (1 or 2 occurrences)

These observations translate the common sense that more frequent TEs are easier to extract because they are supported by more statistical evidence. In this subsection we want to analyse wether the pattern-based extraction also exhibits this trait.

Table 6.10 shows, for each language pair, a series of frequency intervals, the number of extracted TEs that fall within each interval, and the respective precision within the interval. These intervals follow more or less an exponential growth which provides finer detail for lower frequencies.

The well known Zipfian frequency distribution of words [119, 120] says, in broad terms, that most of the distinct phrases occurring in a corpus are infrequent. Interestingly, unlike statistical extraction methods, the number of TEs extracted with the pattern-based method also tends to be higher for lower frequencies, particularly in the case of EN-FR, EN-PT, ES-PT, and FR-PT, where the number of TEs with frequency lower than 2 is greater than the number of TEs in each of the higher-frequency intervals.

Also interestingly, with the exception DE-EN and DE-PT, the precision does not vary drastically between the three lower-frequency intervals, which contain most of the extracted TEs.

Thus, we claim that the precision of the pattern-based extraction is relatively independent of the frequency of extracted TEs.

6.5.6 Precison and Productivity of Individual Translation Patterns

Obviously, not all translation patterns are equally reliable or productive. Much of the merit of the high precision figures presented above should go to the linguists that wrote precise and productive translation patterns.

Is it possible to have highly productive and precise patterns? Or, we must sacrifice one to have the other?

In order to find out if less productive patterns are more precise than more productive ones, we split patterns into two groups based on the number of TEs extracted. We call these groups the *most productive* (MP) and *least productive* (LP).

The splitting was made by sorting patterns based on the number of TEs extracted by each one. Then, starting with the most productive pattern, i.e. the one that extracted the greatest number of TEs, we add patterns one by one to the MP group until the total number of TEs extracted by all patterns already in the MP group is equal or greater to the half the total number of TEs extracted by all patterns.

L1-L2	Frequency	# TEs	% TEs	Precision
CS-PT	1 .. 2	285	11.88%	99.65%
	3 .. 10	767	31.98%	100.00%
	11 .. 100	816	34.03%	100.00%
	101 .. 10k	523	21.81%	100.00%
	10k ..	7	0.29%	100.00%
DE-EN	1 .. 2	10650	39.14%	85.94%
	3 .. 10	11297	41.51%	88.54%
	11 .. 100	4437	16.30%	93.64%
	101 .. 10k	822	3.02%	94.53%
	10k ..	7	0.03%	100.00%
DE-ES	1 .. 2	9761	38.37%	95.56%
	3 .. 10	9768	38.40%	96.69%
	11 .. 100	4713	18.53%	97.92%
	101 .. 10k	1184	4.65%	98.99%
	10k ..	10	0.04%	90.00%
DE-PT	1 .. 2	183901	99.74%	92.46%
	3 .. 10	428	0.23%	97.66%
	11 .. 100	52	0.03%	98.08%
EN-ES*	1 ..	153773	100.00%	93.75%
EN-FR	1 .. 2	48427	31.87%	98.47%
	3 .. 10	56796	37.38%	98.90%
	11 .. 100	37466	24.66%	98.85%
	101 .. 10k	9229	6.07%	98.79%
	10k ..	36	0.02%	91.67%
EN-PT	1 .. 2	252473	56.56%	97.88%
	3 .. 10	137884	30.89%	98.29%
	11 .. 100	51239	11.48%	97.84%
	101 .. 10k	4817	1.08%	92.24%
	10k ..	7	0.00%	42.86%
EN-ZH*	1 ..	1428	100.00%	90.41%
ES-PT	1 .. 2	14464	45.02%	99.43%
	3 .. 10	12907	40.17%	99.29%
	11 .. 100	4289	13.35%	99.00%
	101 .. 10k	470	1.46%	98.51%
FR-PT	1 .. 2	19642	40.66%	96.80%
	3 .. 10	18578	38.45%	98.10%
	11 .. 100	8364	17.31%	99.07%
	101 .. 10k	1719	3.56%	99.36%
	10k ..	10	0.02%	100.00%

Table 6.10: Precision of patterns vs TE frequency. For the language pairs marked with *, the frequency information was lost and thus frequency intervals are not presented.

L1-L2	# Pats	% Pats	# TEs	% TEs	Precision
CS-PT	5	7.58%	1204	50.21%	100.00%
	61	92.42%	1194	49.79%	99.92%
DE-EN	3	8.57%	16854	61.93%	90.73%
	32	91.43%	10359	38.07%	84.97%
DE-ES	3	4.35%	14103	55.45%	99.28%
	66	95.65%	11333	44.55%	93.25%
DE-PT	46	2.18%	92688	50.27%	92.28%
	2061	97.82%	91693	49.73%	92.67%
EN-ES	13	2.16%	79152	51.47%	95.46%
	590	97.84%	74621	48.53%	91.93%
EN-FR	3	1.79%	85895	56.53%	99.72%
	165	98.21%	66059	43.47%	97.46%
EN-PT	16	0.60%	228645	51.22%	98.63%
	2631	99.40%	217775	48.78%	97.21%
EN-ZH	2	11.76%	983	68.84%	90.13%
	15	88.24%	445	31.16%	91.01%
ES-PT	9	9.89%	16737	52.09%	99.67%
	82	90.11%	15393	47.91%	98.90%
FR-PT	12	6.98%	24884	51.51%	97.13%
	160	93.02%	23429	48.49%	98.48%

Table 6.11: Precision of patterns vs number of extracted TEs per pattern.

Thus, both groups contain approximately, but not exactly, the same number of TEs.

Table 6.11 shows the precision of MP and LP groups for each language pair. For each language pair, the top row corresponds to the MP group and the bottom row to the LP group. For most language pairs, the precision of the MP patterns is higher than the precision of the LP patterns. Thus, we can answer the question formulated above, and say that patterns can be simultaneously precise and productive. Case in point, the 3 EN-FR most productive patterns extracted nearly 86 thousand TE with precision above 99%.

6.6 Summary

In this chapter I proposed a language for expressing translation patterns and a method for matching these patterns.

An important feature of the pattern language is the ability to restrict the matching of variables based on word morphology, specifically word suffixes and prefixes.

Another important feature is the ability to specify context restrictions, which require not only that the phrases to be extracted are matched but also some context around them, making the extraction context dependent.

As far as I know, neither of these two features has been previously proposed by other authors.

Compared to statistical extraction methods, such as Anymalign [60], the proposed extraction method has the advantage of being able to extract rare TEs without losing precision. Results also show that patterns can be simultaneously precise and productive.

Chapter Seven

Conclusions

Let us begin the conclusions by taking an overview of the problems addressed in this thesis and how the inner chapters relate to each other, highlighting cross-cutting aspects of the problems and the proposed solutions. The goal of this overview is to bring forward the aspects that bind together the five inner chapters of this thesis.

In Chapters 2 to 4 I have addressed three alignment problems that constitute a pipeline of progressive alignment refinement, starting with an unorganized multilingual collection of documents and ending with aligned phrases, within aligned sentences, in turn within aligned documents.

Besides the affinity of these problems in terms of their inputs and outputs, the proposed solutions are all based on a common *coverage-maximization* approach, that we will revisit later. Thus, one might say that the Chapters 2 to 4 present adaptations of the same fundamental solution to each of the three alignment problems.

In Chapter 5, we move on to the cognate extraction problem where the focus is placed on identifying specific sub-word patterns in human-validated cognates and exploit these patterns for extracting more cognates. These sub-word patterns inspired the development of a more general translation pattern language, in Chapter 6, which enables extraction of word and multi-word translations by relying (heavily) on the phrase alignment method proposed earlier in Chapter 4.

All the relations described above are depicted more succinctly in Figure 7.1, where each chapter is represented as a numbered rectangle.

The bilingual lexicon, placed at the center of the figure, is used as a knowledge source by all methods proposed. Although there is no direct arrow connecting the lexicon to the rectangle of Chapter 6, the lexicon is used to generate phrase alignment candidates which are in turn used to instantiate the variables of translation patterns, as depicted at the bottom-right of the figure.

Therefore, all methods proposed were designed to take advantage of the knowledge contained in a bilingual lexicon, and contribute directly or indirectly to augment the lexicon, which was the driving goal of this thesis.

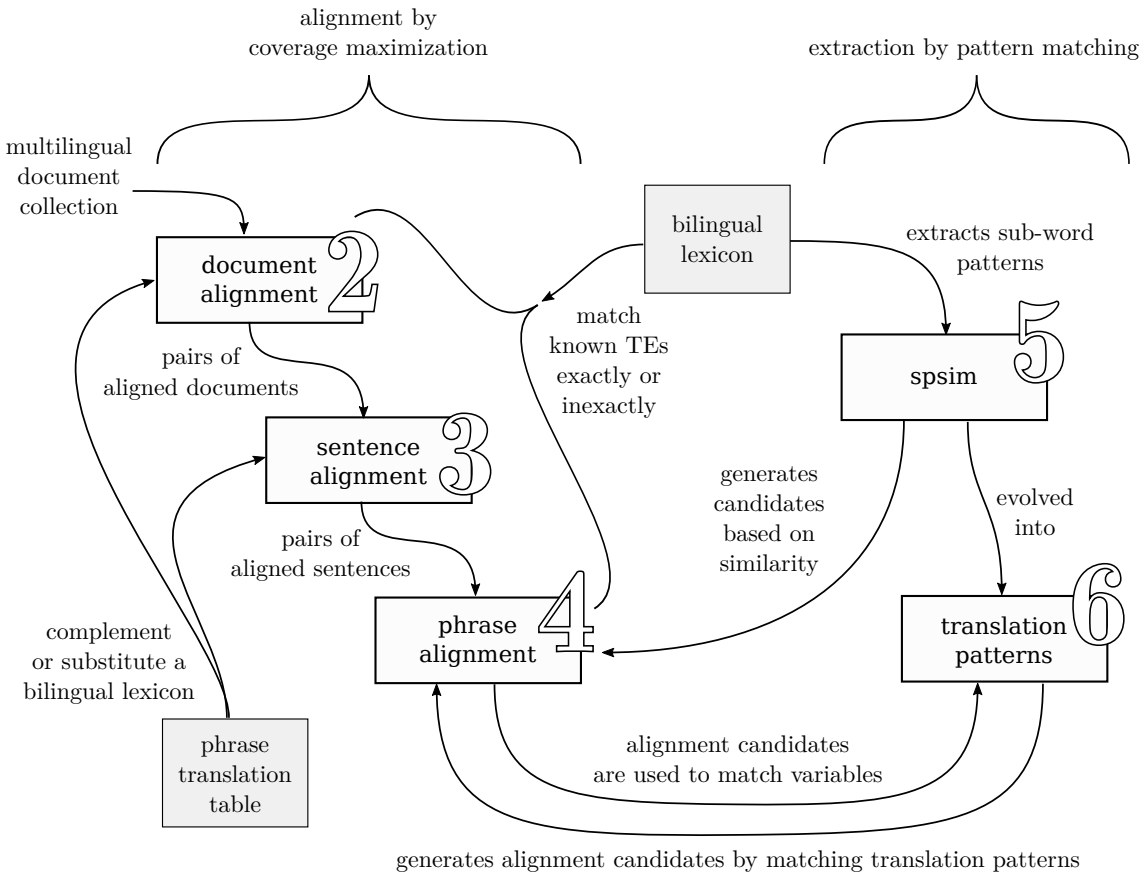


Figure 7.1: Relations between main content chapters of this thesis.

7.1 Findings

In this section I present my interpretation of the results obtained in the evaluation sections of each inner chapter of the thesis.

7.1.1 Regarding Coverage-Based Document and Sentence Alignment

Considering the three alignment problems addressed in this thesis, document and sentence alignment are perhaps the two problems where my proposed methods are more directly comparable to other state-of-the-art methods. I base this statement on two facts:

1. The inputs and outputs of these two problems are clearly identified and there is no leeway for interpretation. By contrast, the phrase alignment problem is subject to various interpretations, discussed further ahead.
2. Besides my document alignment method, the other top-ranked competitors [18, 29] at the WMT16 bilingual document alignment shared task [17] presented solutions that take advantage of previously acquired translation knowledge via machine translation. Similarly, the Bleualign [99] sentence alignment method, which served as basis for implementing my coverage-based sentence aligner, also takes advantage of previously acquired knowledge via machine translation.

I see two main advantages of my coverage-based method compared to machine translation-based methods:

1. The coverage-based methods do not require a translation engine to be executed during alignment, which is itself a computationally costly process;
2. The resulting alignments, at least in these evaluation testsets, have higher quality, perhaps because of errors introduced in the machine translation step.

For document alignment, the use of a Moses phrase translation table enabled marginally better alignments than using our human validated bilingual lexicon, as discussed in Section 2.3. Thus, while a phrase translation table is much noisier than our lexicon, it is also much larger, which apparently is more important than being error free, for this task. For sentence alignment we did not do the same comparison because we do not have a bilingual lexicon for the language pair (DE-FR) of the gold standard evaluation corpus used in the evaluation, but I suspect that the findings will be similar to what we observed in document alignment.

To summarise, I think that the results presented in Chapters 2 and 3 allow us to conclude that the coverage-based approach to alignment is able to produce alignments *at least* as accurate as the alignments produced by all the other methods compared.

Despite the fact that coverage-based alignments have consistently achieved higher precision and recall scores than all the other methods in these evaluations, I think that a bolder claim saying that coverage-based methods are generally more accurate than all other methods needs further evidence from other language pairs and testsets.

7.1.2 Findings Related to Coverage-Based Phrase Alignment

Unlike the document and sentence alignment problems, the phrase alignment problem has many different interpretations in terms of what should be its output, which prevents a direct comparison of different alignment methods. For example, the Anymalign [60] aligner makes alignments internally but outputs only a list of pairs of phrases along with the number of times they were aligned and the respective translation probability scores. In my view, this is a phrase translation extractor, despite the fact that it makes phrase alignments internally. Similarly, the phrase alignment method employed by the Moses toolkit [55] to extract a phrase translation table, also makes alignments internally while working, but never writes them at the output. Instead, it only writes the aligned phrase pairs.

To the best of my knowledge, besides the phrase alignment method proposed in this thesis, no other method produces non-monotonic, hierarchical and discontiguous phrase alignments in place, i.e. making alignments between specific occurrences of phrases in the input sentences.

More important than the output format, there are also different interpretations about the segmentation of alignment units and how they should relate to each other. For example, in the Moses toolkit [55], all contiguous sequences of tokens up to a maximum length

within a sentence are potentially a phrase. In this case, aligned phrases are supposed to overlap with each other. In my view, this is a brute force way of dealing with the problem of segmentation: by extracting all possible segmentations we surely include a correct segmentation, but at the cost of enourmous redundancy. By contrast, the phrase alignment method proposed by Zhang et al [117, 118] has a very strict notion of segmentation: it assumes that there is only one possible segmentation of a sentence into equivalent minimal phrases and that each word can only belong to one phrase. In my view, this segmentation is too strict as it forbids hierarchical phrase alignments such as the ones produced by my coverage-based aligner. For example, if we imagine that Zhang's method aligned "staff training" with "formação de pessoal", then it would not allow the inner alignments of "staff" with "pessoal" and "training" with "formação". Later, in the future work section, I will discuss how these hierarchical alignments could be useful in generating translations for new sentences by reusing as much as possible from similar sentences in a phrase-aligned parallel corpus.

Given the proliferation of interpretations of what should be the output of the phrase alignment problem, it is understandable that most phrase alignment methods are evaluated only indirectly, by evaluating the quality of a translations produced by a phrase-based statistical machine translation (PBSMT) system based on the phrase alignments.

This is also the case of the method proposed in this thesis. I have compared it with the alignment method proposed in my MSc thesis [39], using the Transtor PBSMT system developed by José Aires in the context of his PhD thesis [4].

Because Transtor was designed to work with monotonic alignments, I could not evaluate the newly proposed method to its full potential. Instead, the new non-monotonic hierarchical alignments had to be converted to the old format, loosing a substantial amount of information in this process. Nevertheless, the results suggest that the precision of alignments has not decreased, compared to the old monotonic method. Thus, I believe that better translation quality will be achieved from these alignments if and when the translator makes full use of the newer alignments. It is important to remind that the translations produced by Transtor are evaluated on average 5.1 BLEU points above Moses, for 8 language pairs[1] in all 16 translation directions, when both are trained on the same corpora and lexica [4].

In our participation on the WMT16 biomedical translation shared task, we obtained 1.8 to 3.3 BLEU points above Moses in the EN-PT language pair, which is lower than we expected, but after a brief analysis of the training data we concluded that it contains an unusual amount of noise compared to other corpora that we have been using, as discussed in Section 4.4. These results suggest that our alignment and translation methods may be more sensitive to noise in the input corpus than Moses.

To my understanding, coverage-based and statistical methods do not invalidate each other. Instead, they complement each other. When a bilingual lexicon is not yet available

[1] EN-DE, EN-FR, EN-ES, EN-PT, DE-PT, ES-PT, FR-PT and DE-ES

or is too small, statistics-based alignment *is* the best option. On the other hand, when we have a bilingual lexicon or a phrase translation table, it is wasteful not to take advantage of these resources.

7.1.3 Findings Concerning Cognate Extraction

The relative complexity of previously proposed language-adaptable spelling similarity measures [10, 57, 95, 106] combined with the succinctness of their descriptions in the respective papers and the lack of available implementations, have prevented adoption of those measures by anyone other than their authors.

The spelling similarity (SpSim) measure, proposed in Chapter 5 provides an average 10% gain in terms of F-measure relative to edit-distance-based similarity (EdSim) and longest common sub-sequence ratio (LCSR) static measures. The method is conceptually simple and its implementation takes less than 100 lines of code.

By making the source code of my implementation freely available [2] and documented, other researchers have had the oportunity to try it out and incorporate this measure on their own work [21, 69, 74].

7.1.4 Findings Concerning Extraction Based on Translation Patterns

From the results presented in Chapter 6 I conclude that a single well written translation pattern can be an effective way of extracting a large number of complex phrase translations that many methods fail or strugle to extract.

As an example, the anymalign [60] method was unable to extract the phrase translation "regarding"↔"que respeita" from the DGT-TM corpus [104], despite the fact that this phrase pair co-occurrs in 997 sentence pairs. By contrast, the translation pattern "$1/ing"↔"que $1/e/a/em/am" was able to extract with 91.9% precision 4,423 correct translation equivalents (TEs) from the same corpus, including the pair "regarding"↔"que respeita". Of the extracted TEs, 54% only co-occur once in the corpus.

Looking at the distribution of the number of extracted TEs per pattern, in Table 6.11, we observe that for most language pairs less than 10 patterns are responsible for extracting more than 50% of TEs. Perhaps surprisingly, the patterns that extract a greater number of TEs also tend to be more precise.

7.2 Contributions

This section summarizes the contributions of this thesis, first at a conceptual level and then at a material level.

[2] https://github.com/luismsgomes/spsim

7.2.1 Contributed Ideas

I consider that the core innovation of my proposed alignment methods is the *coverage-maximization* approach. This approach is radically different from previously proposed methods, which are essentially based on likelihood-maximization of statistical models.

While coverage is calculated in different ways for each of these problems, the fundamental concept is the same: coverage is a score that reflects the amount of knowledge, contained in a lexicon or a phrase translation table, that is consistent with a given alignment hypothesis. The underlying assumption of coverage-maximization alignment is that a correct alignment should in principle be consistent with a greater amount of previously acquired knowledge than an incorrect one.

Another core contributed idea comes partially from Chapter 5, where I addressed spelling similarity, but more strongly from Chapter 6 where I proposed the use of translation patterns for extraction. I think that, besides the usefulness of spsim and the proposed translation patterns for extraction, there is a more abstract idea underlying these solutions, which is that many complex translation problems can be easily expressed and solved with simple *deterministic rules*. These deterministic rules do not have to be hard coded in the program. Instead, they can be automatically extracted from examples, as in the case of SpSim, or they can be written manually in an adequate high-level language.

Today, many papers about various text processing topics just report the findings of coupling an off-the-shelf machine learning algorithm with data from a specific problem instance, sometimes without a deep and critical understanding of what is going on in the process, and without seeking a simpler or more effective solution to the problem. I hope that these two extraction methods demonstrate that it is still possible to solve complex problems with simple solutions and I hope they encourage other people to try simple solutions before reaching to more complex tools. A simple solution does not work necessarily worse than a complex one.

7.2.2 Material Contributions

There are three types of material contributions, which I will describe in turn: publications, generated data and programs.

Publications

The methods proposed in Chapters 2, 3 and 5 have been described in dedicated papers [41, 42, 43] and presented in conferences. The method proposed in Chapter 4 has been described briefly in a co-authored paper [6]. The only method that has not yet given rise to a publication is the method proposed in Chapter 6.

Generated Data

Another type of material contributions are the phrase aligned parallel corpora and the extracted and validated bilingual lexica that have been produced with the help of the alignment and extraction methods proposed in this thesis. In the case of bilingual lexica, most credit for the creation of this resource must be attributed to the team of linguists that performed the manual validation. I can take a small credit for the creation of this resource for having provided the extraction methods.

The phrase aligned corpora and the extracted and validated bilingual lexica have enabled three lines of researh.

The first one, by José Aires in the context of his PhD thesis, on the topic of machine translation [4]. In this line of research we have co-authored a paper, already mentioned above, describing our phrase-based machine translation system which participated on the WMT16 biomedical translation shared task [6]. José Aires has already defended his PhD thesis [4].

The second line, by Jorge Costa in the context of his PhD thesis [23], on the topic of employing compressed text indices and space-efficient data structures to index and query phrase-aligned parallel corpora. In this line of research, for which I have contributed an initial working prototype based on suffix arrays and some ideas for further development, we have co-authored 4 publications [24, 25, 26, 27].

The third line of research, by Kavitha Mahesh in the context of her PhD thesis [69], is focused on learning from extracted and validated bilingual lexica and has two sub-topics: (1) automatic classification of newly extracted TEs by a classifier trained on previously human-validated TEs; and (2) learning sub-word bilingual morphology rules from the validated lexica and generate new translations containing word forms that are missing from the lexicon, either because they were not extracted or because they do not occur in the corpus. In this lexica-centered line of research we have co-authored 7 papers [70, 71, 72, 73, 74, 75, 76]. Kavitha Mahesh has recently defended her PhD thesis [69].

Programs

The last material contributions of this thesis are the programs implementing each of the alignment and extraction methods proposed in this thesis. The implementations of the sentence alignment method[3] and SpSim[4] have been released as open source.

All methods work together to form a pipeline, but they also work separately and may be individually integrated into other pipelines. Overall, this translates into a open, loosely coupled architecture, which is easily extended or changed, because all intermediary file formats are textual and easy to parse programmatically. For example, it is easy to add new extractors or to integrate an unsupervised aligner, which could provide unsupervised

[3] https://github.com/luismsgomes/lrec2016sentalign

[4] https://github.com/luismsgomes/spsim

alignment candidates to complement the candidates generated based on the input lexicon and translation patterns.

7.3 Future Work

In this section I propose several lines of work, which could improve and build on the work presented on this thesis.

7.3.1 Learning from Coverage-Based Phrase Alignments

The phrase alignment method proposed in Chapter 4 generates *partial* alignments, meaning that some words in the input sentences may be left unaligned if they are not covered by the lexicon. The inexact candidate generators, *stem_lookup* and *match_sim*, reduce the number of unaligned words by generalizing information contained in the input lexicon. The first by matching TEs based on their stems and the second by matching cognates. However, some words are still left unaligned, whenever their stems are not in the lexicon and their translations do not have similar spellings.

One possible solution to the problem of reducing unaligned words would be to use the final set of alignment candidates obtained by coverage-maximization as training data to a machine learning algorithm, in order to learn how to predict missing alignment candidates given a set of existing alignment candidates.

In recent years, artificial neural networks (ANNs) have been profusely employed to obtain *word embeddings*, which supposedly encode semantic and syntactic attributes of words in a high-dimensional space [80]. While word embeddings are surely interesting, I find the design of the networks that generate them even more so. In essence, these networks are trained to predict a word given the neighbouring words as input[5].

Drawing inspiration from this design, and assuming that most alignment candidates produced by the coverage-maximization method are correct, we could employ an ANN to predict an alignment candidate given neighbouring candidates as input. To train the ANN, we would take each alignment candidate of the final set of candidates of each pair of sentences and encode it in the output vector of the ANN, while the remainder candidates would be encoded in the input vector. For example, if there are three candidates, c_1, c_2, and c_3 in the final alignment of a pair of sentences, then we would generate three training samples by considering the following input (x_i) and output (y_i) candidates:

[5] However, the intended use of these networks is not for making the predictions that they were trained to do. Instead, the final product are the weight vectors of the *embedding layer*, which are the so called word embeddings.

$$x_1 = c_1, c_2 \qquad y_1 = c_3$$
$$x_2 = c_1, c_3 \qquad y_2 = c_2$$
$$x_3 = c_2, c_3 \qquad y_3 = c_1$$

The main problem to be researched is to find an adequate vector representation of the input candidates, which vary in number, and of the output candidate to be predicted.

Besides the neighbouring candidates, the input vector would also contain features based on phrase lengths, occurrence and co-occurrence frequencies, spelling similarity, and possibly others.

There are two hypotheses that are required for this approach to work. The first is that it should be possible to predict each phrase alignment based on *context features*, such as neighbouring phrases and alignments, combined with *intrinsic features* of the phrases being aligned, such as length, occurrence and co-occurrence frequencies, spelling similarity, etc. The second is that coverage-based alignment should produce only very few unaligned phrases in each sentence, so that neighbouring alignments are available to predict alignments for unaligned phrases.

The first hypothesis seems to be true because statistical alignment methods are based mostly on a co-occurrence frequencies and they are able to predict a large percentage of correct candidates. Thus, by adding context features, the ability to predict should improve. The second hypothesis depends essentially on the size of the input lexicon available.

The total number of different training samples available to train the ANN would be equal to the number of alignment candidates generated by the coverage-maximization method over an entire corpus, which only depends on the size of the corpus. Not all candidates are correct though, but this could work in our favor by avoiding overfitting the neural model.

If sucessful, this would be an "indirectly-supervised learning" approach to the phrase alignment problem, since the training of the ANN would not be based directly on human-labelled data, but instead on alignments produced by the coverage-based method, which in turn are based on a human-verified bilingual lexicon.

7.3.2 Exploiting Hierarchical Phrase Alignments for Machine Translation

Earlier, I claimed that besides having a phrase alignment between multiword phrases such as "staff training"↔"formação de pessoal", it would also be important to store the finer-grained alignments such as "staff"↔"pessoal" and "training"↔"formação" along with the coarser-grained alignment.

To see how these finer alignments are useful, imagine that we want to translate the passage "the crew training programme will be reinforced" that is almost identical to the

passage "the staff training programme will be reinforced" which is in our phrase-aligned corpus . By having the finer-grained alignments, and assuming that we know that "crew" can be translated as "tripulação", we would be able to replace "pessoal" by "tripulação" in this phrase-aligned passage which would result in an accurate translation. By storing alignments of multiple granularities, we will be able to select the most adequate one at translation time. I see a phrase-aligned parallel corpus as a repository of sentence translation patterns, where each phrase alignment candidate is a potential variable that can be replaced as needed to generate a sentence translation.

The approach to machine translation sketched in the example above, where we replaced one word on a base sentence to produce a translation, is called example-based machine translation (EBMT) and was introduced in 1984 by Makoto Nagau [82]. Currently, it is employed in some computer assisted translation (CAT) systems[6], but as a research topic it seems to have lost interest to the PBSMT approach.

In my view, today's availability of better hardware and parallel corpora that are many, many times larger than those available in the eighties and nineties, poses new advantages to the example-based approach over a phrase-based statistical approach.

One of them is that in EBMT we avoid *disassembling* the input sentence as much as possible, and instead we focus on reusing sentences or large passages from the base corpus to generate translations by replacing only short phrases within the base sentences.

By contrast, statistical phrase-based models break up the sentences in the training corpus into phrases of all lengths up to a predefined maximum length. Then, only translations for these phrases are stored in the phrase translation table. The original sentences are lost. As a consequence, if we ask a PBSMT system to translate a long sentence that happens to be in base corpus, it is unlikely that it will be able to recreate the original translation.

By hypothesis, if we do not disassemble the base corpus into small pieces, we will be able to *retrieve* whole sentences that are similar to the one that we need to translate. Likewise, if we do not have to disassemble the sentence that we want to translate, then we may avoid many mistakes that we would make while assembling the target-side sentence.

By combining the phrase alignments produced with the coverage-based aligner with automatically learned morphological rules coming from the line of work of Kavitha Mahesh [69], I hypothesize that we would be able to generate sentence translations by reusing as much as possible of sentences already in the corpus and replacing only the phrases that differ in the source language by equivalent phrases in the target language, while respecting monolingual morphological rules such as gender and number agreement, and bilingual morphological rules such as suffix correspondences for regular verbs such as as "*ed"↔"*eu" in "learned"↔"aprendeu".

Another possible advantage of EBMT versus statistical phrase-based models comes from the fact that, in theory, a EBMT approach would continue to benefit from increasing

[6]For example the Déjà Vu system commercialized by Atril http://www.atril.com/.

the size of its base corpus, because as long as we keep adding unseen sentences to the corpus, we keep increasing the probability of finding a similar sentence in the corpus to the one that we need to translate. By contrast, Turchi et al [109] suggest that "it is unlikely that increasing dataset sizes will result in significant improvements" in the performance of unsupervised phrase-based models.

Obviously, increasing the size of the base corpus of an EBMT system requires efficient data structures and algorithms for supporting storage and retrieval of phrase aligned sentences. Jorge Costa, in the context of his PhD thesis [23], has employed compressed text indices for efficiently storing and querying a phrase-aligned parallel corpus, achieving an overall space reduction up to 70%, compared to the space needed when uncompressed data structures are used. Currently, these data structures are word based, which means that morphological decomposition of words has not been exploited yet. In principle, by exploiting morphological decomposition, further compression will be possible.

In my view, these data structures provide a good starting point for implementing an EBMT translator.

7.3.3 Automatic Generation of Translation Patterns

The high precision and productivity of the pattern-based extraction approach proposed in Chapter 6 motivates future research on the complementary problem of generating translation patterns from a human-validated bilingual lexicon.

This subsection presents a simple method for generating patterns from a bilingual lexicon. This method generates only basic translation patterns, without morphological or context restrictions and thus should be regarded as a starting point for future work. Nevertheless, a preliminary evaluation of extraction based on patterns generated by this simple method reveals surprisingly good precision and are thus highly motivating.

The translation pattern generation starts by employing the phrase alignment method proposed in Chapter 4 to align the lexicon as if it was a corpus of very short sentences. As a result of this alignment step, TEs that are strictly subsumed by other TEs will be aligned hierarchically within them, as shown in the left-hand side of Figure 7.2.

Note that each TE being given as the scope of phrase alignment is itself part of the lexicon being used by the aligner. Thus, there will be one trivial candidate which aligns the whole TE, but this trivial candidate is not useful and we ignore it. These trivial candidates are represented in the figure as dotted lines.

For each phrase-aligned TE that has at least one strictly subsumed candidate, we will generate one or more translation patterns by replacing the segments corresponding to each subsumed candidate by a variable.

By replacing only one candidate at a time we generate patterns with a single variable. If we replace two non-overlapping candidates, then we will generate candidates with two variables, and in principle we could continue until we have replaced all candidates. The more candidates we replace, the more variables a pattern will have and thus more

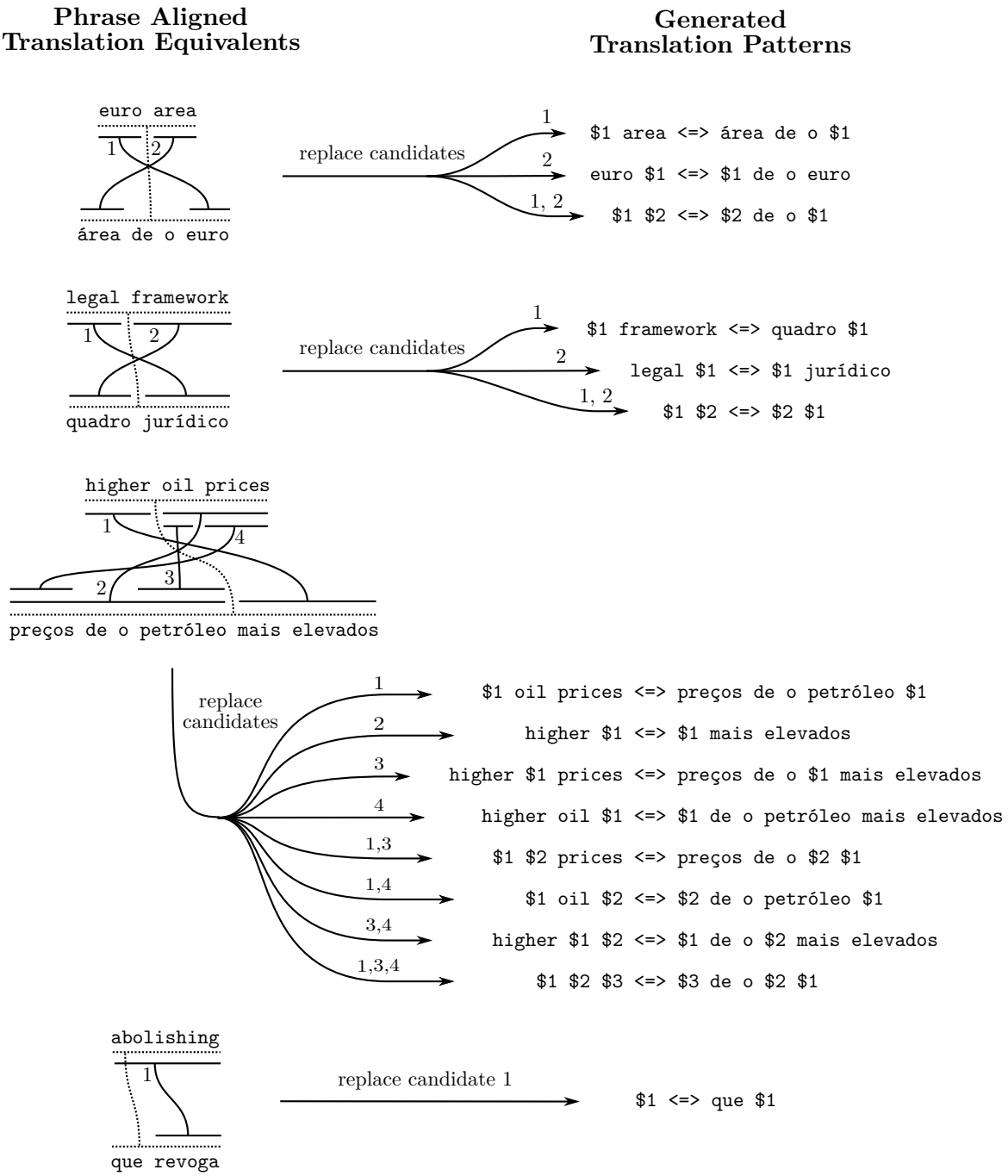


Figure 7.2: Example generation of translation patterns with a single variable.

abstract it will be. Typically, a more abstract pattern will also be more general which means that it will match a greater number of phrase pairs but will be less precise. For the experiments reported below for the EN-PT language pair, we generated patterns with only one variable.

On the left-hand side of Figure 7.2 we have represented the phrase alignments for some TEs of the lexicon and on the right-hand side the resulting translation patterns.

By applying this simple generation method to the the EN-PT bilingual lexicon, containing 1,088,990 TEs, we generated 7,726 single-variable patterns.

By applying these patterns, we were able to extract 38,017 new TEs, of which 24,451 are correct, 10,218 are correct but are monotonically decomposable, and 3,348 are incorrect. The precision of this extraction was 91.19%.

Interestingly, the monotonically decomposable TEs, which are correct but desirable to have in the lexicon, were extracted by patterns that were themselves generated from monotonically decomposable TEs.

Thus, in future experiments, if we want to avoid extracting monotonically decomposable TEs, then we must avoid generating patterns from TEs that are themselves monotonically decomposable. These TEs are easily identifiable from their phrase alignments.

7.3.4 Automatic Generation of Discontiguous Phrases

For each translation pattern generated with the method proposed above, we check if removing the variables results in a TE already present in the lexicon. If it does, then we extract discontiguous phrases by replacing the numbered variables of the pattern by unlinked variables ($*).

For example, using the method above we could obtain the pattern "has $1 updated"↔"$1 actualizou" from "has not updated"↔"não actualizou". If we remove the variables from this pattern, we get the TE "has updated"↔"actualizou", which is also in the lexicon. Therefore, for alignment purposes, we would replace the phrase pair "has updated"↔"actualizou" in the lexicon with the more generic pair, containing a discontiguous phrase, "has $* updated"↔"actualizou".

7.3.5 Automatic Filtering of Aligned Documents and Sentences

Similarly to the classifier that Kavitha Mahesh developed for TEs [74, 76], we could develop a similar classifier for translation memories (TMs), which are pairs of parallel translation units, most commonly sentences. TMs may be produced by CAT tools, during translation, or they can be obtained by sentence-aligning parallel documents, as discussed in Chapter 3.

A binary classifier could be trained to filter out non-parallel TMs from a TM collection, to the benefit of all downstream applications of that collection. This problem has recently gathered interest from the research community, resulting in the first shared task on cleaning of translation memories [9].

The phrase-level alignments within a pair of sentences should be a good indicator of parallelism. If all or nearly all words are aligned, then the TM is almost certainly parallel. However, is there are many words unaligned, then it could be because the sentences are not parallel, but could also be a consequence of the input lexicon not including those words or phrases.

However, if the phrase alignment method is enhanced with machine learning as proposed above, then the number of unaligned words will be further reduced and alignment will become a more reliable indicator of non-parallel sentences as well as parallel ones. A classifier would take as input a set of features derived from alignment candidates and attributes of the sentences, such as lenght proportionality, presence of numbers in each and both sentences, etc.

As training examples we may use TMs that have been manually checked to be parallel as positive examples, and for negative examples we might replace sentences of positive example pairs by similar sentences retrieved from a large monolingual corpora of the corresponding language. Using this method, we know that the positive examples are indeed parallel, since they were checked by hand, but we are unsure if the negative examples are not parallel. Only experimentation will inform us if this method of generating negative examples is good enough. The advantage of this method is that we can generate balanced datasets with as many negative example as positive ones. It is also more realistic than generating negative examples by randomly pairing sentences, because automatic sentence aligners tend to make mistakes when sentences are similar, not when they are randomly dissimilar.

In the same way that phrase alignment can be an indicator of sentence parallelism, sentence alignment can be an indicator of document parallelism. Thus, hypothetically, a classifier could be trained for document pairs and its training data could be generated in a similar way as described above for the TM classifier, but considering documents as units.

Bibliography

[1] José Gabriel Pereira Lopes (PI), Vitor Jorge Ramos Rocio, Luís Manuel Silveira Russo, Joaquim Francisco Ferreira da Silva, Luís Manuel dos Santos Gomes, José Aires Gomes Camacho, and Kavitha Karimbi Mahesh. *ISTRION – Improving Phrase-Based Statistical Machine TRanslation through supervisION*. Funded by FCT/MCTES under contract PTDC/EIA-EIA/114521/2009, Mar. 1, 2011–Feb. 28, 2013.

[2] Sadaf Abdul-Rauf, Mark Fishel, Patrik Lambert, Sandra Noubours, and Rico Sennrich. "Extrinsic evaluation of sentence alignment systems". In: *Proceedings of LREC 2012 Workshop on Creating Cross-language Resources for Disconnected Languages and Styles*. May 2012, pp. 6–10. URL: http://www.lrec-conf.org/proceedings/lrec2012/workshops/26.Credislas-Proceedings.pdf.

[3] Alfred V. Aho and Margaret J. Corasick. "Efficient String Matching: An Aid to Bibliographic Search". In: *Commun. ACM* 18.6 (June 1975), pp. 333–340. ISSN: 0001-0782. DOI: 10.1145/360825.360855.

[4] José Aires. "Genuine phrase-based statistical machine translation with supervision". PhD thesis. Monte da Caparica: Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (FCT-UNL), 2015. URL: https://run.unl.pt/handle/10362/18153.

[5] José Aires, Gabriel Pereira Lopes, and Luís Gomes. "Phrase Translation Extraction from Aligned Parallel Corpora using Suffix Arrays and Related Structures". In: *Progress in Artificial Intelligence — 14th Portuguese Conference on Artificial Intelligence, EPIA 2009*. Aveiro, Portugal: Springer, Oct. 2009, pp. 587–597. DOI: 10.1007/978-3-642-04686-5_48.

[6] José Aires, Gabriel Lopes, and Luís Gomes. "English-Portuguese Biomedical Translation Task Using a Genuine Phrase-Based Statistical Machine Translation Approach". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 456–462. DOI: 10.18653/v1/W16-2335. URL: http://www.aclweb.org/anthology/W/W16/W16-2335.

[7] Andoni Azpeitia and Thierry Etchegoyhen. "DOCAL - Vicomtech's Participation in the WMT16 Shared Task on Bilingual Document Alignment". In: *Proceedings*

of the First Conference on Machine Translation. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 666–671. URL: http://www.aclweb.org/anthology/W/W16/W16-2364.pdf.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2014). URL: http://arxiv.org/abs/1409.0473.

[9] Eduard Barbu, Carla Parra Escartín, Luisa Bentivogli, Matteo Negri, Marco Turchi, Constantin Orasan, and Marcello Federico. "The First Automatic Translation Memory Cleaning Shared Task". In: *Machine Translation* 30.3-4 (Dec. 2016), pp. 145–166. ISSN: 0922-6567. DOI: 10.1007/s10590-016-9183-x.

[10] S. Bergsma and G. Kondrak. "Alignment-based discriminative string similarity". In: *Annual Meeting – Association for Computational Linguistics*. Vol. 45. 2007, p. 656.

[11] Michael W. Berry and Paul G. Young. "Using latent semantic indexing for multilanguage information retrieval". In: *Computers and the Humanities* 29.6 (1995), pp. 413–429. ISSN: 1572-8412. DOI: 10.1007/BF01829874.

[12] Victor Bilbao, Gabriel Pereira Lopes, and Tiago Ildefonso. "Measuring the impact of cognates in parallel text alignment". In: *2005: Portuguese Conference on Artificial Intelligence, Proceedings*. Ed. by Amilcar Cardoso Carlos Bento and Gael Dias. IEEE Computer Society, Dec. 2005, pp. 338–343.

[13] Ondřej Bojar et al. "Findings of the 2016 Conference on Machine Translation". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 131–198. URL: http://www.aclweb.org/anthology/W/W16/W16-2301.

[14] Fabienne Braune and Alexander Fraser. "Improved Unsupervised Sentence Alignment for Symmetrical and Asymmetrical Parallel Corpora". In: *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. COLING '10. Beijing, China: Association for Computational Linguistics, 2010, pp. 81–89. URL: http://dl.acm.org/citation.cfm?id=1944566.1944576.

[15] Peter F. Brown, Jennifer C. Lai, and Robert L. Mercer. "Aligning Sentences in Parallel Corpora". In: *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*. ACL '91. Berkeley, California: Association for Computational Linguistics, 1991, pp. 169–176. DOI: 10.3115/981344.981366.

[16] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. "The Mathematics of Statistical Machine Translation: Parameter Estimation". In: *Computational Linguistics* 19.2 (June 1993), pp. 263–311. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=972470.972474.

[17] Christian Buck and Philipp Koehn. "Findings of the WMT 2016 Bilingual Document Alignment Shared Task". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 554–563. URL: http://www.aclweb.org/anthology/W/W16/W16-2347.pdf.

[18] Christian Buck and Philipp Koehn. "Quick and Reliable Document Alignment via TF/IDF-weighted Cosine Distance". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 672–678. URL: http://www.aclweb.org/anthology/W/W16/W16-2365.pdf.

[19] David Chiang. "A Hierarchical Phrase-based Model for Statistical Machine Translation". In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. ACL '05. Ann Arbor, Michigan: Association for Computational Linguistics, 2005, pp. 263–270. DOI: 10.3115/1219840.1219873.

[20] David Chiang. "Hierarchical Phrase-Based Translation". In: *Computational Linguistics* 33.2 (June 2007), pp. 201–228. ISSN: 0891-2017. DOI: 10.1162/coli.2007.33.2.201.

[21] Alina Maria Ciobanu and Liviu P. Dinu. "Automatic Detection of Cognates Using Orthographic Alignment". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 99–105. URL: http://www.aclweb.org/anthology/P14-2017.

[22] Corinna Cortes and Vladimir Vapnik. "Support Vector Machine". In: *Machine learning* 20.3 (1995), pp. 273–297.

[23] Jorge Costa. "Time and Space Efficient Data Structures for Supporting Machine Translation Tasks". PhD thesis. Monte da Caparica: Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (FCT-UNL), 2017.

[24] Jorge Costa, Luís Gomes, Gabriel Pereira Lopes, and Luís Russo. "Improving Bilingual Search Performance using Compact Full-Text Indices". In: *Computational Linguistics and Intelligent Text Processing — 16th International Conference, CICLing 2015*. Cairo, Egypt: Springer, Apr. 2015, pp. 582–595. DOI: 10.1007/978-3-319-18111-0_44.

[25] Jorge Costa, Luís Gomes, Gabriel Pereira Lopes, and Luís Russo. "Representing a Bilingual Lexicon with Suffix Trees". In: *Proceedings of the 2011 ACM Symposium on Applied Computing*. Taichung, Taiwan: ACM, Mar. 2011, pp. 1164–1165. ISBN: 978-1-4503-0113-8. DOI: 10.1145/1982185.1982440.

[26] Jorge Costa, Luís Gomes, Gabriel Pereira Lopes, Luís Russo, and Nieves Brisaboa. "Compact and Fast Indexes for Translation Related Tasks". In: *Progress in Artificial Intelligence — 16th Portuguese Conference on Artificial Intelligence, EPIA 2013*. Angra do Heroísmo, Azores, Portugal: Springer, Sept. 2013, pp. 504–515. ISBN: 978-3-642-40668-3. DOI: 10.1007/978-3-642-40669-0_43.

[27] Jorge Costa, Gabriel Pereira Lopes, Luís Gomes, and Luís Russo. "Managing and Querying a Bilingual Lexicon with Suffix Trees". In: *15th Portuguese Conference on Artificial Intelligence, EPIA 2011*. Lisbon, Portugal, Oct. 2011.

[28] I. Dagan, K. Church, and W. Gale. "Robust Bilingual Word Alignment for Machine Aided Translation". In: *Natural Language Processing Using Very Large Corpora*. Ed. by Susan Armstrong, Kenneth Church, Pierre Isabelle, Sandra Manzi, Evelyne Tzoukermann, and David Yarowsky. Dordrecht: Springer Netherlands, 1999, pp. 209–224. ISBN: 978-94-017-2390-9. DOI: 10.1007/978-94-017-2390-9_13.

[29] Aswarth Abhilash Dara and Yiu-Chang Lin. "YODA System for WMT16 Shared Task: Bilingual Document Alignment". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 679–684. URL: http://www.aclweb.org/anthology/W/W16/W16-2366.pdf.

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1 (1977), pp. 1–38. ISSN: 00359246. URL: http://www.jstor.org/stable/2984875.

[31] John DeNero, Dan Gillick, James Zhang, and Dan Klein. "Why Generative Phrase Models Underperform Surface Heuristics". In: *Proceedings on the Workshop on Statistical Machine Translation*. New York City: Association for Computational Linguistics, June 2006, pp. 31–38. URL: http://www.aclweb.org/anthology/W/W06/W06-3105.

[32] John DeNero and Dan Klein. "Discriminative Modeling of Extraction Sets for Machine Translation". In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. ACL '10. Uppsala, Sweden: Association for Computational Linguistics, 2010, pp. 1453–1463. URL: http://dl.acm.org/citation.cfm?id=1858681.1858828.

[33] Gaël Harry Adélio André Dias Dias. "Extraction automatique d´associations lexicales à partir de corpora". PhD thesis. Lisboa: Universidade Nova de Lisboa, 2002.

[34] Miquel Esplà-Gomis, Mikel Forcada, Sergio Ortiz Rojas, and Jorge Ferrández-Tordera. "Bitextor's participation in WMT'16: shared task on document alignment". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 685–691. URL: http://www.aclweb.org/anthology/W/W16/W16-2367.pdf.

[35] Miquel Esplà-Gomis, Felipe Sánchez-Martínez, and Mikel L. Forcada. "Combining Content-Based and URL-Based Heuristics to Harvest Aligned Bitexts from Multilingual Sites with Bitextor". In: *The Prague Bulletin of Mathemathical Linguistics* 93 (2010), pp. 77–86. URL: http://www.mt-archive.info/MTMarathon-2010-Espla-Gomis.pdf.

[36] Kevin Flanagan. "Bilingual phrase-to-phrase alignment for arbitrarily-small datasets". In: *AMTA 2014: proceedings of the eleventh conference of the Association for Machine Translation in the Americas, Vancouver, BC, October 22-26*. 2014, pp. 83–95. URL: http://www.mt-archive.info/10/AMTA-2014--Flanagan.pdf.

[37] William A. Gale and Kenneth W. Church. "A Program for Aligning Sentences in Bilingual Corpora". In: *Computational Linguistics* 19.1 (Mar. 1993), pp. 75–102. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=972450.972455.

[38] Ulrich Germann. "Bilingual Document Alignment with Latent Semantic Indexing". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 692–696. URL: http://www.aclweb.org/anthology/W/W16/W16-2368.pdf.

[39] Luís Gomes. "Parallel Texts Alignment". MA thesis. Monte da Caparica: Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (FCT-UNL), 2009. URL: http://hdl.handle.net/10362/2051.

[40] Luís Gomes, José Aires, and Gabriel Pereira Lopes. "Parallel Texts Alignment". In: *New Trends in Artificial Intelligence — 14th Portuguese Conference on Artificial Intelligence, EPIA 2009*. Aveiro, Portugal: Universidade de Aveiro, Oct. 2009, pp. 513–524. ISBN: 978-972-96895-4-3. URL: http://epia2009.web.ua.pt/onlineEdition/513.pdf.

[41] Luís Gomes and Gabriel Pereira Lopes. "First Steps Towards Coverage-Based Document Alignment". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 697–702. DOI: 10.18653/v1/W16-2369. URL: http://www.aclweb.org/anthology/W/W16/W16-2369.pdf.

[42] Luís Gomes and Gabriel Pereira Lopes. "First Steps Towards Coverage-Based Sentence Alignment". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016. ISBN: 978-2-9517408-9-1. URL: http://www.lrec-conf.org/proceedings/lrec2016/pdf/959_Paper.pdf.

[43] Luís Gomes and Gabriel Pereira Lopes. "Measuring Spelling Similarity for Cognate Identification". In: *Progress in Artificial Intelligence — 15th Portuguese Conference on Artificial Intelligence, EPIA 2011*. Lisbon, Portugal: Springer, Oct. 2011, pp. 624–633. ISBN: 978-3-642-24768-2. DOI: 10.1007/978-3-642-24769-9_45.

[44] Miquel Esplà Gomis, Felipe Sánchez Martínez, and Mikel L. Forcada. "A Simple Approach to Use Bilingual Information Sources for Word Alignment". In: *Procesamiento de Lenguaje Natural* 49 (2012). ISSN: 1989-7553. URL: http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/4552.

[45] Dan Gusfield. *Algorithms on Strings, Trees and Sequences – Computer Science and Computational Biology*. Cambridge University Press, 1997.

[46] Stéphane Huet and Philippe Langlais. "Translation of Idiomatic Expressions Across Different Languages: A Study of the Effectiveness of TransSearch". In: *Where Humans Meet Machines: Innovative Solutions for Knotty Natural-Language Problems*. Ed. by Amy Neustein and Judith A. Markowitz. New York, NY: Springer New York, 2013, pp. 185–209. ISBN: 978-1-4614-6934-6. DOI: 10.1007/978-1-4614-6934-6_9. URL: https://doi.org/10.1007/978-1-4614-6934-6_9.

[47] John Hutchins. "From First Conception to First Demonstration: the Nascent Years of Machine Translation, 1947–1954. A Chronology". In: *Machine Translation* 12.3 (1997), pp. 195–252. ISSN: 1573-0573. DOI: 10.1023/A:1007969630568.

[48] Tiago Ildefonso and Gabriel Pereira Lopes. "Longest Sorted Sequence Algorithm for Parallel Text Alignment". In: *Computer Aided Systems Theory – EUROCAST 2005* (2005), pp. 81–90.

[49] Pierre Isabelle, Marc Dymetman, George Foster, Jean-Marc Jutras, Elliott Macklovitch, Francois Perrault, Xiaobo Ren, and Michel Simard. "Translation Analysis and Translation Automation". In: *Proceedings of the 1993 Conference of the Centre for Advanced Studies on Collaborative Research: Distributed Computing - Volume 2*. CASCON '93. Toronto, Ontario, Canada: IBM Press, 1993, pp. 1133–1147. URL: http://dl.acm.org/citation.cfm?id=962367.962416.

[50] *ISTRION-BOX - Translation & Revision, Lda*. URL: http://istrionbox.com/.

[51] Laurent Jakubina and Phillippe Langlais. "BAD LUC@WMT 2016: a Bilingual Document Alignment Platform Based on Lucene". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 703–709. URL: http://www.aclweb.org/anthology/W/W16/W16-2370.pdf.

[52] Xiaoyi Ma Kazuaki Maeda and Stephanie Strassel. "Creating Sentence-Aligned Parallel Text Corpora from a Large Archive of Potential Parallel Text using BITS and Champollion". In: *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. Marrakech, Morocco: European Language Resources Association (ELRA), May 2008. ISBN: 2-9517408-4-0. URL: http://www.lrec-conf.org/proceedings/lrec2008/pdf/779_paper.pdf.

[53] Philipp Koehn. "Europarl: A parallel corpus for statistical machine translation". In: *The Tenth Machine Translation Summit*. Vol. 5. 2005, pp. 79–86. URL: http://www.mt-archive.info/MTS-2005-Koehn.pdf.

[54] Philipp Koehn. *Statistical machine translation*. Cambridge University Press, 2010.

[55] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. "Moses: Open Source Toolkit for Statistical Machine Translation". In: *Proceedings of the*

45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 177–180. URL: http://www.aclweb.org/anthology/P07-2045.

[56] Philipp Koehn, Franz Josef Och, and Daniel Marcu. "Statistical Phrase Based Translation". In: *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*. 2003. URL: http://acl.ldc.upenn.edu/N/N03/N03-1017.pdf.

[57] G. Kondrak. "Identification of Cognates and Recurrent Sound Correspondences in Word Lists". In: *Traitement Automatique des Langues* 50.2 (2009), pp. 201–235.

[58] Grzegorz Kondrak and Tarek Sherif. "Evaluation of Several Phonetic Similarity Algorithms on the Task of Cognate Identification". In: *Proceedings of the Workshop on Linguistic Distances*. LD '06. Sydney, Australia: Association for Computational Linguistics, 2006, pp. 43–50. ISBN: 1-932432-83-3. URL: http://dl.acm.org/citation.cfm?id=1641976.1641983.

[59] Fethi Lamraoui and Philippe Langlais. "Yet another fast, robust and open source sentence aligner. time to reconsider sentence alignment". In: *Proceedings of the XIV Machine Translation Summit* (2013), pp. 77–84.

[60] Adrien Lardilleux and Yves Lepage. "A truly multilingual, high coverage, accurate, yet simple, subsentential alignment method". In: *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas (AMTA)*. Waikiki, Hawaii, 2008, pp. 125–132. URL: http://www.mt-archive.info/AMTA-2008-Lardilleux.pdf.

[61] Thanh Le, Hoa Trong Vu, Jonathan Oberländer, and Ondřej Bojar. "Using Term Position Similarity and Language Modeling for Bilingual Document Alignment". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 710–716. URL: http://www.aclweb.org/anthology/W/W16/W16-2371.pdf.

[62] Vladimir I. Levenshtein. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals". In: *Soviet Physics Doklady* 10 (Feb. 1966), pp. 707–710.

[63] Peng Li, Maosong Sun, and Ping Xue. "Fast-Champollion: A Fast and Robust Sentence Alignment Algorithm". In: *Coling 2010: Posters*. Beijing, China: Coling 2010 Organizing Committee, Aug. 2010, pp. 710–718. URL: http://www.aclweb.org/anthology/C10-2081.

[64] Percy Liang, Ben Taskar, and Dan Klein. "Alignment by agreement". In: *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics. 2006, pp. 104–111.

[65] Pintu Lohar, Haithem Afli, Chao-Hong Liu, and Andy Way. "The ADAPT Bilingual Document Alignment system at WMT16". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 717–723. URL: http://www.aclweb.org/anthology/W/W16/W16-2372.pdf.

[66] Xiaoyi Ma. "Champollion: A robust parallel text sentence aligner". In: *LREC 2006: Fifth International Conference on Language Resources and Evaluation*. 2006, pp. 489–492. URL: http://www.lrec-conf.org/proceedings/lrec2006/pdf/746_pdf.pdf.

[67] Xiaoyi Ma and Mark Liberman. "BITS: A Method for Bilingual Text Search over the Web". In: *Machine Translation Summit VII, Singapore*. Sept. 1999, pp. 538–542. URL: http://www.mt-archive.info/MTS-1999-Ma-2.pdf.

[68] Sainik Mahata, Dipankar Das, and Santanu Pal. "WMT2016: A Hybrid Approach to Bilingual Document Alignment". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 724–727. URL: http://www.aclweb.org/anthology/W/W16/W16-2373.pdf.

[69] Kavitha Karimbi Mahesh. "Augmenting Translation Lexica by Learning Generalised Translation Patterns". PhD thesis. Monte da Caparica: Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (FCT-UNL), 2017. URL: http://hdl.handle.net/10362/21995.

[70] Kavitha Karimbi Mahesh, Luís Gomes, and José Gabriel Pereira Lopes. "Using Bilingual Segments in Generating Word-to-word Translations". In: *Proceedings of the Sixth Workshop on Hybrid Approaches to Translation (HyTra-6)*. Osaka, Japan: The Association for Computational Linguistics, Aug. 2016. URL: http://glicom.upf.edu/hytra6/pdf/HyTra608.pdf.

[71] Kavitha Mahesh, Luís Gomes, José Aires, and Gabriel Pereira Lopes. "Classification and Selection of Translation Candidates for Parallel Corpora Alignment". In: *Progress in Artificial Intelligence — 17th Portuguese Conference on Artificial Intelligence, EPIA 2015*. Coimbra, Portugal: Springer, Sept. 2015. ISBN: 978-3-319-23485-4. DOI: 10.1007/978-3-319-23485-4_73.

[72] Kavitha Mahesh, Luís Gomes, and Gabriel Pereira Lopes. "Bilingually motivated segmentation and generation of word translations using relatively small translation data sets". In: *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation (PACLIC 29)*. Shanghai, China, Oct. 2015. URL: http://aclweb.org/anthology/Y/Y15/Y15-2022.pdf.

[73] Kavitha Mahesh, Luís Gomes, and Gabriel Pereira Lopes. "Identification of Bilingual Segments for Translation Generation". In: *Advances in Intelligent Data Analysis XIII — 13th International Symposium, IDA 2014*. Leuven, Belgium: Springer, Oct. 2014, pp. 167–178. DOI: 10.1007/978-3-319-12571-8_15.

[74] Kavitha Mahesh, Luís Gomes, and Gabriel Pereira Lopes. "Identification of Bilingual Suffix Classes for Classification and Translation Generation". In: *Advances in Artificial Intelligence — 14th Ibero-American Conference on AI, IBERAMIA 2014*. Santiago de Chile, Chile: Springer, Nov. 2014, pp. 154–166. DOI: 10.1007/978-3-319-12027-0_13.

[75] Kavitha Mahesh, Luís Gomes, and Gabriel Pereira Lopes. "Learning Clusters of Bilingual Suffixes using Bilingual Translation Lexicon". In: *Mining Intelligence and Knowledge Exploration*. Lecture Notes in Computer Science. Springer International Publishing, Dec. 2015. DOI: 10.1007/978-3-319-26832-3_57.

[76] Kavitha Mahesh, Luís Gomes, and Gabriel Pereira Lopes. "Using SVMs for Filtering Translation Tables for Parallel Corpora Alignment". In: *15th Portuguese Conference on Artificial Intelligence, EPIA 2011*. Lisbon, Portugal, Oct. 2011. ISBN: 978-989-95618-4-7.

[77] Daniel Marcu and William Wong. "A Phrase-based, Joint Probability Model for Statistical Machine Translation". In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 133–139. DOI: 10.3115/1118693.1118711.

[78] Marek Medved, Miloš Jakubícek, and Vojtech Kovár. "English-French Document Alignment Based on Keywords and Statistical Translation". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 728–732. URL: http://www.aclweb.org/anthology/W/W16/W16-2374.pdf.

[79] Dan Melamed I. "Bitext maps and alignment via pattern recognition". In: *Computational Linguistics* 25.1 (1999), pp. 107–130.

[80] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. "Linguistic Regularities in Continuous Space Word Representations". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 746–751. URL: http://www.aclweb.org/anthology/N13-1090.

[81] Robert C. Moore. "Fast and Accurate Sentence Alignment of Bilingual Corpora". English. In: *Machine Translation: From Research to Real Users*. Ed. by StephenD. Richardson. Vol. 2499. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, pp. 135–144. ISBN: 978-3-540-44282-0. DOI: 10.1007/3-540-45820-4_14.

[82] Makoto Nagao. "A Framework of a Mechanical Translation Between Japanese and English by Analogy Principle". In: *Proc. Of the International NATO Symposium on Artificial and Human Intelligence*. Lyon, France: Elsevier North-Holland, Inc.,

1984, pp. 173–180. ISBN: 0-444-86545-4. URL: http://dl.acm.org/citation.cfm?id=2927.2938.

[83] Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. "An Unsupervised Model for Joint Phrase Alignment and Extraction". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. HLT '11. Portland, Oregon: Association for Computational Linguistics, 2011, pp. 632–641. ISBN: 978-1-932432-87-9. URL: http://dl.acm.org/citation.cfm?id=2002472.2002553.

[84] Franz Josef Och. "An Efficient Method for Determining Bilingual Word Classes". In: *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*. EACL '99. Bergen, Norway: Association for Computational Linguistics, 1999, pp. 71–76. DOI: 10.3115/977035.977046.

[85] Franz Josef Och and Hermann Ney. "The Alignment Template Approach to Statistical Machine Translation". In: *Computational Linguistics* 30.4 (Dec. 2004), pp. 417–449. ISSN: 0891-2017. DOI: 10.1162/0891201042544884.

[86] Franz Josef Och, Christoph Tillmann, and Hermann Ney. "Improved Alignment Models for Statistical Machine Translation". In: *Proceedings of the Joint Conference of Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP-VLC)*. 1999, pp. 20–28. URL: http://acl.ldc.upenn.edu/W/W99/W99-0604.pdf.

[87] Vassilis Papavassiliou, Prokopis Prokopidis, and Stelios Piperidis. "The ILSP/ARC submission to the WMT 2016 Bilingual Document Alignment Shared Task". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 733–739. URL: http://www.aclweb.org/anthology/W/W16/W16-2375.pdf.

[88] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135.

[89] Martin F. Porter. "Readings in Information Retrieval". In: ed. by Karen Sparck Jones and Peter Willett. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997. Chap. An Algorithm for Suffix Stripping, pp. 313–316. ISBN: 1-55860-454-5. URL: http://dl.acm.org/citation.cfm?id=275537.275705.

[90] Martin F. Porter. *Snowball: A language for stemming algorithms*. Oct. 2001. URL: http://snowball.tartarus.org/texts/introduction.html.

[91] Alexandre Rafalovitch and Robert Dale. "United Nations General Assembly Resolutions: A Six-Language Parallel Corpus". In: *Proceedings of the MT Summit XII*. International Association of Machine Translation. Ottawa, Canada, Aug. 2009, pp. 292–299.

[92] Philip Resnik. "Parallel Strands: A Preliminary Investigation into Mining the Web for Bilingual Text". In: *Machine Translation and the Information Soup: Third Conference of the Association for Machine Translation in the Americas AMTA'98 Langhorne, PA, USA, October 28–31, 1998 Proceedings*. Ed. by David Farwell, Laurie Gerber, and Eduard Hovy. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 72–82. ISBN: 978-3-540-49478-2. DOI: 10.1007/3-540-49478-2_7.

[93] Philip Resnik and Noah A. Smith. "The Web As a Parallel Corpus". In: *Computational Linguistics* 29.3 (Sept. 2003), pp. 349–380. ISSN: 0891-2017. DOI: 10.1162/089120103322711578.

[94] António Ribeiro. "Parallel Texts Alignment for Extraction of Translation Equivalents". PhD thesis. Lisboa: Universidade Nova de Lisboa, 2002.

[95] António Ribeiro, Gael Dias, G. P. Lopes, and João Tiago Mexia. "Cognates Alignment". In: *Proceedings of the Machine Translation Summit VIII (MT Summit VIII), Santiago de Compostela, Spain, September 18-22, 2001*. Ed. by Bente Maegaard. European Association of Machine Translation, Sept. 2001, pp. 287–292.

[96] António Ribeiro, Gabriel Pereira Lopes, and João Mexia. "Extracting Equivalents from Aligned Parallel Texts: Comparison of Measures of Similarity". In: *Advances in Artificial Intelligence: International Joint Conference 7th Ibero-American Conference on AI 15th Brazilian Symposium on AI IBERAMIA-SBIA 2000 Atibaia, SP, Brazil, November 19–22, 2000 Proceedings*. Ed. by Maria Carolina Monard and Jaime Simão Sichman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 339–349. ISBN: 978-3-540-44399-5. DOI: 10.1007/3-540-44399-1_35.

[97] Marina Sanchez-Torron and Philipp Koehn. "Machine Translation Quality and Post-Editor Productivity". In: *AMTA 2016, Vol.* (2016), p. 16.

[98] Jacques Savoy. "Light Stemming Approaches for the French, Portuguese, German and Hungarian Languages". In: *Proceedings of the 2006 ACM Symposium on Applied Computing*. SAC '06. Dijon, France: ACM, 2006, pp. 1031–1035. ISBN: 1-59593-108-2. DOI: 10.1145/1141277.1141523.

[99] Rico Sennrich and Martin Volk. "MT-based sentence alignment for OCR-generated parallel texts". In: *The Ninth Conference of the Association for Machine Translation in the Americas (AMTA 2010), Denver, Colorado*. 2010. URL: http://mt-archive.info/AMTA-2010-Sennrich.pdf.

[100] Vadim Shchukin, Dmitry Khristich, and Irina Galinskaya. "Word Clustering Approach to Bilingual Document Alignment (WMT 2016 Shared Task)". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 740–744. URL: http://www.aclweb.org/anthology/W/W16/W16-2376.pdf.

[101] M Simard, G Foster, and P Isabelle. "Using cognates to align sentences in parallel corpora". In: *In Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation*. 1992, pp. 67–81.

[102] Pankaj Singh, Ashish Kulkarni, Himanshu Ojha, Vishwajeet Kumar, and Ganesh Ramakrishnan. "Building Compact Lexicons for Cross-Domain SMT by Mining Near-Optimal Pattern Sets". In: *Advances in Knowledge Discovery and Data Mining: 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19-22, 2016, Proceedings, Part I*. Ed. by James Bailey, Latifur Khan, Takashi Washio, Gill Dobbie, Joshua Zhexue Huang, and Ruili Wang. Cham: Springer International Publishing, 2016, pp. 290–303. ISBN: 978-3-319-31753-3. DOI: 10.1007/978-3-319-31753-3_24.

[103] Jason R. Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. "Dirt Cheap Web-Scale Parallel Text from the Common Crawl". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 1374–1383. URL: http://www.aclweb.org/anthology/P13-1135.

[104] Ralf Steinberger, Andreas Eisele, Szymon Klocek, Spyridon Pilos, and Patrick Schlüter. "DGT-TM: A freely available Translation Memory in 22 languages". In: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Ed. by Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis. Istanbul, Turkey: European Language Resources Association (ELRA), May 23–25, 2012. ISBN: 978-2-9517408-7-7.

[105] George Tambouratzis, Fotini Simistira, Sokratis Sofianopoulos, Nikos Tsimboukakis, and Marina Vassiliou. "A resource-light phrase scheme for language-portable MT". In: *Proceedings of the 15th International Conference of the European Association for Machine Translation, Leuven, Belgium*. May 2011, pp. 185–192.

[106] J. Tiedemann. "Automatic construction of weighted string similarity measures". In: *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. 1999, pp. 213–219.

[107] Jörg Tiedemann. "Parallel Data, Tools and Interfaces in OPUS". In: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Ed. by Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis. Istanbul, Turkey: European Language Resources Association (ELRA), May 23–25, 2012. ISBN: 978-2-9517408-7-7.

[108] Christoph Tillmann. "A Projection Extension Algorithm for Statistical Machine Translation". In: *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*. EMNLP '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 1–8. DOI: 10.3115/1119355.1119356.

[109] Marco Turchi, Tijl De Bie, and Nello Cristianini. "Learning Performance of a Machine Translation System: a Statistical and Computational Analysis". In: *Proceedings of the Third Workshop on Statistical Machine Translation*. Columbus, Ohio: Association for Computational Linguistics, June 2008, pp. 35–43. URL: http://www.aclweb.org/anthology/W/W08/W08-0305.

[110] Dániel Varga, Péter Halácsy, András Kornai, Viktor Nagy, László Németh, and Viktor Trón. "Parallel corpora for medium density languages". In: *Recent Advances in Natural Language Processing IV: Selected papers from RANLP 2005*. 2007, pp. 247–258. DOI: 10.1075/cilt.292.32var.

[111] Ashish Venugopal, Stephan Vogel, and Alex Waibel. "Effective Phrase Translation Extraction from Alignment Models". In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. ACL '03. Sapporo, Japan: Association for Computational Linguistics, 2003, pp. 319–326. DOI: 10.3115/1075096.1075137.

[112] Iñaki San Vicente and Iker Manterola. "PaCo2: A Fully Automated tool for gathering Parallel Corpora from the Web". In: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), May 2012. ISBN: 978-2-9517408-7-7. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/231_Paper.pdf.

[113] Stephan Vogel, Hermann Ney, and Christoph Tillmann. "HMM-based Word Alignment in Statistical Translation". In: *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*. COLING '96. Copenhagen, Denmark: Association for Computational Linguistics, 1996, pp. 836–841. DOI: 10.3115/993268.993313.

[114] Xiaolin Wang, Masao Utiyama, Andrew Finch, Taro Watanabe, and Eiichiro Sumita. "Leave-one-out Word Alignment without Garbage Collector Effects". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1817–1827. URL: http://aclweb.org/anthology/D15-1209.

[115] Zdenek Zabokrtsky, Jan Ptacek, and Petr Pajas. "TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer". In: *Proceedings of the Third Workshop on Statistical Machine Translation*. Columbus, Ohio: Association for Computational Linguistics, June 2008, pp. 167–170. URL: http://www.aclweb.org/anthology/W/W08/W08-0325.

[116] Richard Zens, Franz Josef Och, and Hermann Ney. "Phrase-Based Statistical Machine Translation". In: *Proceedings of the German Conference on Artificial Intelligence (KI 2002)*. 2002. URL: http://www.rzens.com/Zens_KI_2002.pdf.

[117] Y. Zhang, S. Vogel, and A. Waibel. "Integrated phrase segmentation and alignment algorithm for statistical machine translation". In: *Proc. Int Conf. Natural Language Processing and Knowledge Engineering*. Oct. 2003, pp. 567–573. DOI: 10.1109/NLPKE.2003.1275970.

[118] Ying Zhang and Stephan Vogel. "Competitive Grouping in Integrated Phrase Segmentation and Alignment Model". In: *Proceedings of the ACL Workshop on Building and Using Parallel Texts*. ParaText '05. Ann Arbor, Michigan: Association for Computational Linguistics, 2005, pp. 159–162. URL: http://dl.acm.org/citation.cfm?id=1654449.1654484.

[119] George Kingsley Zipf. *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio Books, 2016.

[120] George Kingsley Zipf. "The psycho-biology of language." In: (1935).

Appendix A

The Aho-Corasick Automaton

The Aho-Corasick (AC) algorithm [3] is used in the phrase alignment method proposed in Chapter 4. While this algorithm is often described and implemented to operate on character strings, it may be applied to sequences of any type of symbol. In this thesis we applied AC to sequences of tokens, word stems, and characters.

The fundamental problem that this algorithm solves is the following: assume that we have a *large* set of sequences with arbitrary lengths, known apriori. For example, this could be the set of all unique English phrases for which we know at least one Portuguese translation. Now, suppose that we are given an English text, also with arbitrary length, and we need to find, quickly, *all occurrences* of all aforementioned English phrases within the text.

The AC algorithm constructs a minimal deterministic finite automaton (DFA) which in turn allows us to solve this problem with optimal time complexity. A minimal DFA is the smallest automaton (with the least number of states) that can recognize a given set of sequences. Note that in the working example, the whole set of English phrases may be much larger than the text or vice versa, depending if we are going to use our automaton to process a single sentence or a whole corpus. More importantly, note that the set of phrases of interest is known apriori, but the texts where we will search for occurrences of these phrases are not. Therefore, we use AC to pre-process the set of phrases *once*, and then we have an automaton that allows us to search for occurrences of these phrases in any text of any length, very efficiently. Periodically, we update the automaton to incorporate new phrases that have been extracted and validated. The period between updates may range from a few minutes to several days, depending on the rate at which new phrases are extracted and validated. The construction of the automaton takes only a few seconds for a set with 10^6 phrases.

The AC automaton is based on a graph structure called a *keyword tree*, which we will introduce next.

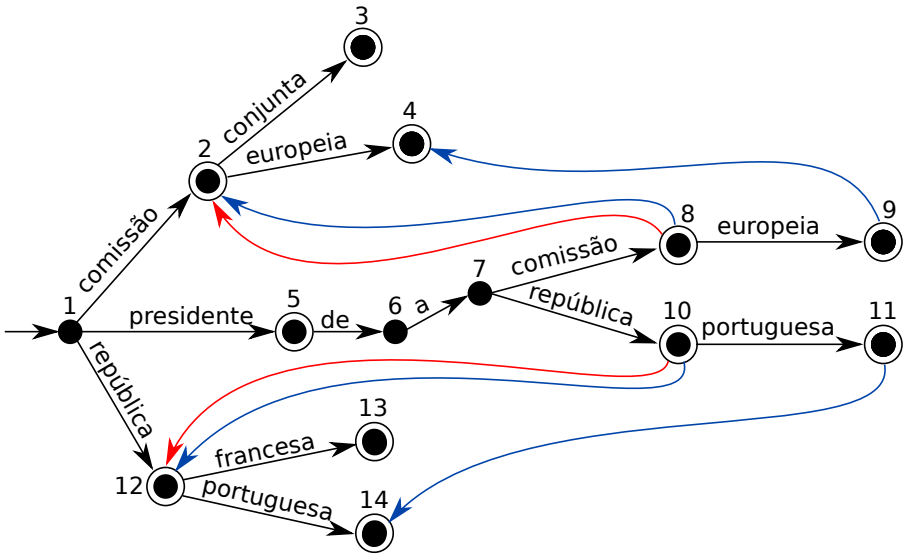


Figure A.1: Example Keyword Tree

A.1 Keyword Tree

A *keyword tree* is a tree-shaped directional graph where the edges are labeled with symbols. The set of sequences of interest, also called *keywords*, are encoded as paths over the graph, starting at the root and ending at marked *output nodes*.

An example tree is given in Figure A.1. This is a word-based tree, meaning that edges are labeled with words. In the Figure, output nodes are enclosed in outer circles to distinguish them from non-output nodes. For example, node 5 is an output node while node 6 is not. This tree encodes 11 distinct phrases, corresponding to the paths running from the root node to each of the output nodes:

- "comissão" (node 2)
- "comissão conjunta" (node 3)
- "comissão europeia" (node 4)
- "presidente" (node 5)
- "presidente de a comissão" (node 8)
- "presidente de a comissão europeia" (node 9)
- "presidente de a república" (node 10)
- "presidente de a república portuguesa" (node 11)
- "república" (node 12)
- "república francesa" (node 13)
- "república portuguesa" (node 14)

A.2 Searching the Tree

The fundamental matching operation of a keyword tree with respect to an input sequence is performed by consuming one symbol at a time from the input and attempting to *descend* one node at a time, starting at the root node, and always following an edge with a symbol that is identical to the current input symbol. For example, let us assume that our input sequence is "presidente de a comissão". We initialize our *current node* pointer to point to the root node 1. Next we take the first symbol from the input sequence, which is "presidente", and we check if there are any outbound edges labeled with that symbol. Since there is one outbound edge labeled "presidente", we follow it and we update the *current node* pointer to point to node 5. We have thus *descended* in our keyword tree. Next, we get the next symbol from the input, which is "de", and we attempt to descend from the current node using this symbol (word). We repeat these steps until either the input sequence has been exhausted (no more symbols) or it is not possible to descend from the current node with the current symbol. At the end of our example input we would have descended to node 8.

Keyword trees are often employed to implement *associative maps* whenever the association keys are sequences. In such scenario, the matching operation is performed as described above and if after the last input symbol has been consumed the current node is an output node, then the value associated with the sequence is returned, else null is returned indicating that the input sequence is not associated with any value.

A.3 Failure and Output Links

As said earlier, the Aho-Corasick automaton is based on a keyword-tree. Descending the tree by matching input symbols to the edge labels, as we saw above, mimics an automaton. The edges are analogous to transitions and the nodes to states. However, such an automaton is *incomplete* because not all symbols are present as labels of outbound edges of all nodes. To be complete, an automaton must provide transitions for each symbol for each state.

The Aho-Corasick algorithm turns a keyword tree into a complete automaton, by augmenting it with *failure* and *output links*.

Failure and output links are special edges in the tree structure. To avoid confusion with the symbol-labeled edges that we have been using thus far, we call them links.

The *failure link* of a node is followed whenever we are unable to descend from that node with the next input symbol, i.e. the node does not have any child nodes accessible through edges labeled with the next input symbol. In the example of Figure A.1, let us assume that we have successfully descended along the path "o presidente de a república", arriving at node 10, and the next input symbol is the word "francesa". Since we cannot descend from the current node with this symbol, we follow the failure link of the node, marked in red, and we reach node 12 with path "república". Now we try to descend

from this node with the symbol "francesa" and we succeed, arriving at node 13 with path "república francesa". This small example tree only has two failure nodes represented: from node 8 to node 2, with paths "presidente de a comissão" and "comissão", respectively, and from node 10 to node 12, with paths "presidente de a república" and "república", respectively. Although ommitted from the drawing, all other nodes have failure links pointing to the root node.

The *output link* of a node is always followed whenever we reach that node, unless the link is null. Output links are necessary to report matches of phrases that are proper suffixes of a node path. For example, let us imagine that we have descended along the path "presidente de a comissão" and we reached node 8. Since the current node is an *output node*, we report a match of the phrase "presidente de a comissão" starting at current input text position minus 4 tokens (the depth of node 8). However, this node also has an output link, which is drawn in blue, pointing at node 2. Therefore, we save aside the current node and then we follow the output link into node 2. We report a match of the phrase "comissão" at the current input text position minus one token (the depth of node 2) and we try to follow an output link from this node. Since node 2 does not have an output link, we cannot follow it, and therefore we have finished reporting matches. We recover the node that we saved aside before we started following output links, node 8, and we resume our tree-descending algorithm by consuming another input symbol and trying to follow an outgoing edge from this node, and so on and so forth.

Appendix B

A Short Introduction To Regexes

The name *regular expression*, derives from the fact that regular expressions define *regular languages* which, in formal language theory, are the simplest type of languages. However, today's implementations of regular-expression matching have features that far exceed the expressive power of regular languages. There are two such features, *capture groups* and *back-references*, that are fundamental for the implementation of our pattern-based extractor described in Chapter 6. Because an expression with back-references does not describe a regular language, it is not a *regular expression* in the formal sense. Hence, to avoid confusion and incorrect use of the term *regular expression*, we will hereafter use the term *regex* instead.

Next, we will learn the most basic regex concepts, and then we will learn about *capture groups* and *back-references*.

B.1 Basic Regex Features

A regex is composed of characters that represent themselves (literals) and characters or sub-expressions that have special meaning. We will not be exhaustive because this appendix is not meant to be a complete reference to regexes, but we will learn about the most important special characters and sub-expressions.

It is also worth mentioning that the syntax shown here is the one used by the Python *regex* module available on Pypi[1]

B.1.1 Character Classes and the Dot Character

A character class matches a single character if and only if the character belongs to the specified class. For example, the regex `\d` will match any digit character and `\s` will match any whitespace character (space, tab, etc). There are many pre-defined character classes and we may also define our own classes. To define a character class, we list all the

[1] The *regex* module is a more featureful alternative to the built-in *re* module and is available at `https://pypi.python.org/pypi/regex/`.

characters that should belong to the class within square brackets. For example, the regex `[0123456789]` matches any single digit.

The hyphen/minus character (`-`) allows us to specify a range of characters. For example, the previous regex could be rewritten more succintly as `[0-9]`. If we wish to include the hyphen/minus character as a member of the class, then it must be placed as the first character after the opening square bracket. For example, the regex `[-+]` matches either a minus or a plus sign.

The dot character (`.`) matches a single arbitrary[2] character. For example, the regex `a.c` matches any 3-character string that begins with an `a` and ends with a `c`. Thus, it would match the strings `abc`, `a/c`, `a4c` or `a.c`, but not the strings `ac` or `abbc`.

B.1.2 Quantifiers

The asterisc (`*`) is a *quantifier* operator that represents a contiguous sequence of zero or more occurrences of the preceeding character or sub-expression. For example, the regex `[0-9]*` matches sequences of digits (possibly empty).

The plus sign (`+`) is another *quantifier* operator which indicates that the preceeding character or sub-expression must appear at least once and then may repeat itself any number of times. For example, the regex `[0-9]+` matches integer numbers with any number of digits (at least one).

The question mark (`?`) is yet another *quantifier* operator which indicates that the preceeding character or sub-expression may appear once or not at all. For example, the regex `[-+]?[0-9]+` matches integer numbers preceeded with an optional minus or plus sign.

B.1.3 Greedy vs Reluctant vs Possessive Quantifiers

The quantifiers introduced thus far are *greedy* quantifiers, meaning that the regex matching engine will initially try to match as many characters from the input text as possible. If the whole regex cannot be matched, then the engine will attempt increasingly shorter matches for each sub-expression affected by a greedy quantifier.

For example, consider the regex `a*a`. When we try to match this regex to the string `aaa`, the engine will greedily match the `a*` to as many characters as possible. Therefore, the engine will *consume* the entire string, `aaa`, because `a*` can match the entire string. However, we only matched the regex partially. To find a full regex match, the engine needs to try again, this time matching the `a*` to one-fewer characters than in the previous attempt. Therefore, this time `a*` will match the first two `a`s in the string (`aa`a) and the second `a` in the regex will be matched to the third `a` in the string (`aa`a), which will result in a full match of the regex.

[2] By default, the dot matches any character except the newline character, but most regex engines have an option to enable the dot to match newlines as well.

The process of trying multiple combinations of sub-expression matches in order to match the whole regex is called *backtracking*.

A *reluctant* quantifier has the opposite behaviour of a *greedy* one: the engine will initially match as few characters from the input as possible and will only try increasingly longer matches for each sub-expression affected by a reluctant quantifier as needed until the whole regex is matched. A reluctant quantifier is defined by inserting a question mark (`?`) next to an asterisc or plus operator. For example, when searching for all matches of regex `a*?a` in the input string `aaa`, we will get a total of 6 matches for three different sub-strings (`a`, `aa`, `aaa`) as follows:

- Sub-string `a` is matched by the whole regex when `a*` matches the empty string. This sub-string is found at three places in the input string: `aaa`, `aaa` and `aaa`;
- Sub-string `aa` is matched by the whole regex when `a*` matches a single `a`. This sub-string is found at two places in the input string: `aaa` and `aaa`;
- Sub-string `aaa` is matched by the whole regex when `a*` matches `aa`.

There is a third type of quantifiers, called *possessive* quantifiers. Like greedy quantifiers, these will match as many characters from the input text as possible. However, unlike greedy quantifiers, the engine will *not* attempt shorter matches. Possessive quantifiers are specified by appending a plus sign (`+`) to a greedy quantifier. For example, the regex `a*+.` will match as many `a`s from the beginning of the input text as possible. This regex will match the string `aab` but not the string `aaa` because in the later case the possessive quantifier will consume all `a`s from the input and leaves no character to match the final dot in the regex.

B.1.4 Escaping

The backslash character (`\`) is used to remove (escape) the special meaning of the character to its right. For example, if we want to match an asterisc in the text then we would prepend it with a backslash (`\*`) to avoid the special meaning of the asterisc as a quantifier.

B.1.5 Boundaries

There are four boundary conditions that are frequently used in regexes:

- `\b` matches an empty string at the begining or end of a word, but not within;
- `\B` matches an empty string in the opposite conditions (within a word but not at the beginning or end);
- `\A` or `^` matches the beginning of the input text line;
- `\Z` or `$` matches the end of the input text line.

For example, regex `\bun\B` will match the string `un` only when it appears as the prefix of a word. Another example, regex `\A\s*\Z` will match empty lines or lines that only contain whitespace.

B.1.6 Disjunctions

A disjunction (or alternation) of several regexes is defined by separating them with a vertical bar (`|`). For example, the regex `a*b|b*a` will match either

1. zero or more `a`s followed by a `b`; or
2. zero or more `b`s followed by an `a`.

B.1.7 Grouping

Parentheses allow us to group sub-expressions. A group restricts the scope of the disjunction operator, which is useful to define *disjunctions* within a larger regex. For example, the regex `x(a*b|b*a)` matches sequences starting with `x` followed by either a sequence of `a`s followed by a `b` or a sequence of `b`s followed by an `a`, and terminated with a `y`. A group also allows a quantifier operator to be applied to the whole regex contained within the group. For example, the regex `x(ab)+y` will match sequences starting with a `x`, followed by `ab` repeated one or more times, and terminated with a `y`.

B.2 Capture Groups

There are three types of groups:

1. *unnamed capture groups*, which are defined just with a pair parentheses, as in the regex `x(a*b|b*a)y`;
2. *named capture groups*, which are defined by inserting `?P<name>` right after the opening parenthesis, replacing `name` with the intended group name, as in the regex `x(?P<sub>a*b|b*a)y` which defines a capture group named *sub*; and
3. *non-capturing groups*, which are defined by inserting `?:` right after the opening parenthesis of a group, as in the regex `x(?:a*b|b*a)y`.

The text matched by each capture group in a regex may be accessed separately. For example, when matching the regex `x(?P<sub>aa|bb)y`, we would be able to get the text that was matched by the grouped expression and thus know whether `aa` or `bb` was matched. If a regex does not have capture groups, then we only have access to the text that matches the whole regex. Groups are numbered from left to right according to the position of their opening parenthesis. The group with the leftmost opening parenthesis will be group 1, the one with the second leftmost opening parenthesis will be group 2, and so on. To access the text matched by unnamed capture groups we must know their number. By contrast, named capture groups may be accessed either by name or by number. For example, the regexes `x(aa|bb)y` and `x(?P<inner>aa|bb)y` are identical except that the first one defines an unnamed capture group while the second defines a group named *inner*.

B.3 Back-References

Back-references are references to the text that was matched by a capture group defined somewhere to the left of the back-reference in the regex. A back-reference will be matched only by the exact same text that matched the referenced capture group. For an example, let us consider the regex `<(\d+)>.+?</\1>`. The parentheses define a unnamed capture group which will match an integer number (one or more digits) enclosed within angle brackets, `<` and `>`. The sub-expression `\1` is a *back-reference* to the first unnamed group defined in the regex (in this case there is only one). As a consequence, this regex will match an arbitrary integer number enclosed within angle brackets, such as for example `<123>` followed by an arbitrary string with at least one character, followed by the exact same integer preceeded by a slash and enclosed within angle brackets, like `</123>`.

For named capture groups, we may create back-references using the syntax `(?P=name)`, replacing `name` with the target group name. For long and complex regexes, such as the ones employed by the pattern-based extractor proposed in Chapter 6, named capture groups and back-references greatly improve readability.

B.4 Summary

In this appendix we learned the basics of *regexes*, *capture groups* and *back-references*.

To consolidate our understanding of these concepts, let us look at an example regex that employs all of them:

```
<(?P<num>\d+)>.+?</(?P=num)>
```

Also, consider the following example input text which has a phrase enclosed within tags[3] `<123>` and `</123>`:

```
The <123>quick brown fox</123> jumps over the lazy dog.
```

On Figure B.1 we show a match of the regex above to this text, and we employ blue and pink rectangles to break down the pattern and the matched text into pieces for discussion.

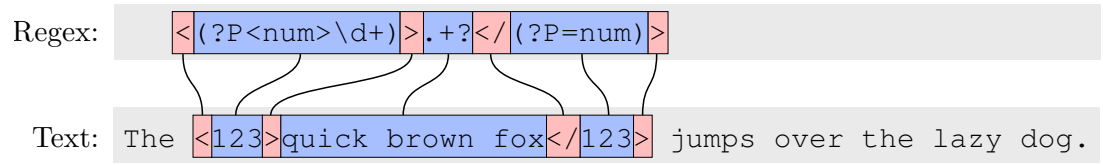


Figure B.1: Example matching of a regex with back-references.

Pink rectangles contain characters that match themselves (literals) and blue rectangles enclose sub-expressions that we discuss next, one by one. Within the leftmost blue

[3] This type of tags is employed in the pattern-based extractor presented in Chapter 6

rectangle of the pattern we define a capture group named *num*. Inside this capture group we have the regex `\d+` which matches a sequence of one or more digits (i.e. an unsigned integer number). In this case, this capture group matches `123`.

In the second blue rectangle we have the regex `.+?` which matches a sequence of one or more arbitrary characters. Note that we employed here a reluctant quantifier, which means that this regex will match the shortest sequence needed to enable a match of the full regex.

Finally, the last blue rectangle we have a back-reference to the capture group *num*, which will match the exact same number that was matched by the regex within the referred group (in this example it was `123`). This concludes our discussion about this example.

Many programming languages have *regex matching* implementations readily available in their standard libraries (Java, Python, Perl, etc) but there is some variation in the features available and syntax adopted by each implementation. The syntax presented in this appendix is the one adopted by the Python *regex* module[4]. A more in-depth tutorial about Python regexes can be found in `https://docs.python.org/3/howto/regex.html`.

[4] `https://pypi.python.org/pypi/regex/`